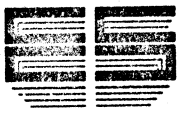
EVANS & SUTHERLAND COMPUTER CORPORATION

CHANNEL CONTROL MAINTENANCE MANUAL

February 17, 1970



Evans & Sutherland Computer Corporation
Three Research Road
Salt Lake City, Utah 84112

CHANNEL CONTROL MAINTENANCE MANUAL

Table of Contents

INTRODUCTION

The maintenance manual for the Channel Control or processor, as it is most often referred to, was written in an attempt to explain the electrical and logical function of the Channel Control. The main emphasis is on the logic function of the processor and a fair amount of detail is included that deals directly with the logical functions of each of the cards that make up the processor. The intent is to provide the reader with sufficient detail to allow him to be able to maintain and trouble shoot the system.

It should be noted that the manual is more of a guide to documentation than documentation in and of itself. For this reason, it is imperative, for any real understanding of the processor and its functions, that the references given in the manual be carefully studied as the manual is studied. In case of conflict between the two, the source documents are naturally to be believed rather than the manual.

The manual assumes the reader has some sort of working knowledge of the functions of TTL level digital circuitry of which the process is composed. It is also assumed that the reader is familiar with the overall purpose and function of the E & S Line Drawing System.

A fairly detailed description of the algorithm is given first, followed by the descriptions of the individual cards in the processor. Important source documents defining the

algorithm and read only memory bits are included in an appendix at the end of the manual.

NUMBERING AND NOMENCLATURE CONVENTIONS

2.1 Numbering

Three groups of numbers are especially significant for the documentation package. The first group identifies the parts, subassemblies, and assemblies used to build the system. The second group refers to physical locations on the back panel or on the component boards. The numbers used to identify different sections of the manuals make up the third group.

2.1.1 E & S Part Numbers

Each part used in the system has been assigned an E & S Part Number. These part numbers are related to part descriptions and vendor numbers by the E & S Guzzinta Chart. With one exception, these numbers have no significance other than to identify the actual part used.

The exception is in the case of the E & S part numbers assigned to integrated circuits. These numbers are six-digit numbers with a three-digit dash number appended. The first three digits are 807; the fourth indicates the number of pins on the IC involved; the fifth and the sixth identify the type of IC (2-wide AND-OR-Invert, etc.). The three-digit dash number relates to tolerances in speed, input load, and drive factor. These numbers allow the substitution of different IC's provided that they have the same logical function and pin numbering and meet the tolerance established by the dash number. These numbers

decode as follows:

807NID-SLD where

N = Number of pins

2 = 24-28 pins
3 = 36 pins
4 = 14 pins
6 = 16 pins

ID = Identification number attached to the IC (usually relates to a vendor number)

S = Speed (odd numbers indicate fixed speed; even numbers indicate minimum speed allowable)

L = Input load in ma

D = Drive factor in ma/10

S	L	D
0-1 Open	0 = 0.0	0 = 6.0
2-3 74HTTL (5-7ns)	1 = 0.4	1 = 5.4
4-5 Signetics (8-10ns)	2 = 0.8	2 = 4.8
6-7 Medium 74TTL (12-14ns)	3 = 1.2	3 = 2.8
	4 = 1.6	4 = 2.4
	5 = 2.0	5 = 2.0
	6 = 2.4	6 = 1.6
	7 = 2.8	7 = 1.2
	8 = 3.2	8 = 0.8
	9 = 3.6	9 = 0.4

2.1.2 Location Numbers

Back panel positions are identified by card slot numbers and connector numbers. The card slot numbers are shown in Figure 4.1.

Test points and connectors (pins for intercard connections) have been numbered as labeled on the boards. Test points run from 1-18 on the solder side of the board and have letter codes from A to V (not continuous) on the component side. Test points on the component side are labeled by letter on the logic drawings and etchings, but identified by numbers on the wirelist. The correspondance is as follows:

A = 19	H = 25	P = 31
B = 20	J = 26	R = 32
C = 21	K = 27	S = 33
D = 22	L = 28	T = 34
E = 23	M = 29	U = 35
F = 24	N = 30	V = 36

The connectors are numbered from 1-98. The even numbers are on the solder side of the board and the odd on the component side. Connectors 1 and 97 carry Vcc and 2 and 98 carry ground.

The boards used contain 70 bug (IC) positions that are numbered from 10-79. The tens figure identifies the column and the ones figure identifies the row of the bug position.

2.1.3 Manual Numbering

This manual is numbered by sections. Page numbers for each section as well as subsection numbers and figure numbers are prefaced by the number of the section.

2.2 Notations

An effort has been made to keep the nomenclature and notations as consistant as possible throughout the manual. However, a few of the conventions used and not used require a little explanation.

2.2.1 Philosophy of Wire and Card Naming

The names attached to the individual wires and the different cards of the system are somewhat arbitrary, but an attempt has been made to make the name some sort of mnemonic indicating the purpose of the wire or board in question.

Wires on the back panel have names that are not necessarily the same as the name of the signals on the cards. Also, a signal may have one name on one card and another on a different card. This often necessitates the use of the backpanel wirelist when tracing a signal from card to card.

Signals which require a low pulse to enable the desired function, or data signals which are in compliment form, are denoted by the "not" symbols. In the manual and throughout the wire lists, these "not" signals are denoted by an *. On the logic diagrams, the symbol used may be either an ∿ or a superscripted line (e.g., $\overline{\text{CLRAIC}}$).

2.2.2 Capitalization

Wire names are given in upper case letters. If a data bus or a group of signals is referenced, only the bus name or the common part of the signal name is given. This was done to avoid awkward repetitions. For example, the signals *ZINT (0) - *ZINT (16) are referred to simply as the *ZINT bus, and COUNT (Q1), COUNT (Q2), COUNT (Q3), COUNT (Q4) are simply referred to as COUNT.

When a reference to a signal's function is made, the name of the function may not be capitalized even though it corresponds in part with the signal's name. Thus, signals controlling "carry ins" for an adder are sometimes called "carry in signals" and not CARRY IN signals, even though the actual name of the signals may be CARRY IN A, CARRY IN B, etc.

Card names are capitalized as proper names in contrast to

the complete capitalization of the wire names. Thus, the "DONE" signal comes from the "Done card."

CHANNEL CONTROL ALGORITHM

3.1 Introduction

The Channel Control is a computer which uses PDP-10 memory and contains an instruction repertoire designed to simplify the programming required for displays. The Channel Control, which is often referred to as the "display processor," or simply "processor" supplies the other devices in the E & S Line Drawing System with the data and instructions needed for their operations.

In order to understand the processor algorithm, one should first become familiar with the registers of the processor. These registers and their interconnections are shown on E & S block diagram 101101-900 (Appendix section 20.3). In addition to these registers, the WHO register is referred to in the algorithm descriptions. This 4-bit register, found on the Instruction card, stores the name of the register from which data may be fetched. It is also necessary to refer to the bits of the processor ROM. Appendix 20.5 contains the definition of the ROM bits in terms of PDP-9 macros. Appendix 20.4 contains the resulting print out of the ROM bits and 20.2 the accompanying bit descriptions. The algorithm itself is also defined by PDP-9 macros as contained in Appendix 20.1. Reference to the flow diagram of the algorithm (101140-950) is also essential as one attempts to understand the algorithm and its individual steps.

3.2 Reference

E & S flow diagram No. 101140-950, PDP-10 Channel Algorithm.

E & S block diagram No. 101101-900, PDP-10 Display Processor.

3.3 Algorithm

The four sheets of the processor algorithm flow diagram divide nicely by function. The first sheet covers the algorithm steps that preceed the fetching of instructions and the actual fetching steps. The second sheet covers steps dealing with instruction decoding and the steps required to execute simple instructions. The third sheet deals with the steps required for the more complex drawing instructions. Sheet four shows the steps of the WRITE and DATA FETCH subroutines. These divisions will be roughly followed in the algorithm description.

For each step, a short description of the function of the step and its relation to the total function of the processor will be given. Following this will be a list of the individual ROM bits engaged by the step. The ROM bits fall into 8 groups as shown below:

1. Register Control (R)
2. Path Control (P)
3. Poke Control (K)
4. Flag Control (F)
5. I/O Control (M)
6. Testing Control
7. Jump Control
8. Jump Address

The bits engaged in the first 5 groups will be listed under the heading "ROM BITS." The last 3 groups affect the progression of the program counter and will be listed under the title PC (Program Counter).

The numbers assigned to the steps relate to the corresponding ROM code in the program counter, which drives the ROM word for that step. As the discussion of the steps will follow a sequence chosen to correspond with the flow diagram and logical chain of steps, the steps will not necessarily be discussed in numerical order.

It should also be noted that the two subroutines WRITE and DATA FETCH may be entered from any of several steps. The PC contents for the state from which these subroutines are entered is saved in a register on the Microcode Control card, which allows the subroutines to return this value to the program counter when the subroutine is finished.

3.3.1 Pre-fetch States

On the upper part of sheet 1 of the flow diagram, the steps are shown which are a preliminary to instruction fetching.

(0) CLEA

The CLEA step is only used on a cold start. It clears various flags and indicators to initiate operation and puts the processor in the STOP state so that it will not fetch data immediately.

ROM BITS:	PC:
F Clear mode	Jump STOP
Clear outflag	

Clear execute flag

Clear F1

Clear F2

Clear IOMBR

K PRIV

M Acknowledge write buffer request flag

(10) STOP

The STOP state merely serves as a jump address

ROM BITS:	PC:
None	PC+1

(11) PREP

The PREP state is a "pre-paused" state used to stall the processor until the pipeline is empty. The condition SETTLED indicates that all units of the pipeline have completed their tasks and are awaiting input. If the pipeline is not settled, the WRITE subroutine may be called so that data can be written into memory in an orderly fashion.

ROM BITS:	PC:
None	Hold until SETTLED-DONE and call WRIT if WRITE REQ

(12) PAUS

After the PREP step, the PAUS step is entered and the processor remains in the paused condition until a PAUS REQUEST is received from the parent computer.

ROM BITS:	PC:
Stopped & Ready	Hold until STOP=DONE and jump IOFF if IOMBR else PC+1

(13) MODE

If the instruction is not sent via the I/O bus, the processor will go to the MODE step. The MODE step forms the main return point for beginning all instructions, except those via the I/O bus. In the MODE step, dispatch 0 decodes the mode flip-flops and directs the processor to the appropriate steps.

1. GP3 and SC=0
2. REPT
3. EXEQ
4. PROG or PEEL (mutually exclusive)

The EXEQ, PROG, and PEEL modes require instruction fetches. The GP3 and REPT modes do not, but rather dispatch directly to instruction decoding steps.

ROM BITS:	PC:
F K PRIV	Dispatch Ø unless PAUSE REQUEST or STOP, then PC=1

(14) MODF

MODF is entered from MODE if the pause request flip-flop is set or if the processor is stopped and does nothing more than provide a jump address to return to the STOP state.

ROM BITS:	PC:
None	Jump STOP

3.3.2 Instruction Fetching

There are four ways in which the processor can get its instructions. (1) from the PDP-10 I/O bus (IOFF). (2) from where the program counter register (PC) points (PCFE). (3) from where the stack pointer register (SP) points (SPFE). (4) from where the read address register (RAR) points (RAFE). The mode of the processor determines which of these sequences is initiated. The steps are identical except for the name of the register used in the data fetch. In the last 3 cases, the instruction fetch requires two steps with the later step making the actual memory cycle request. The double lines on the bottom of the rectangles representing those steps indicate that the clock pulse which terminates them will be delayed until the data requested is available in the memory buffer register (MBR).

(1) IOFF

If the instruction is sent directly to the processor via the I/O bus, the processor will go from the PAUS step to IOFF. IOFF loads the data into the MBR and the left half of the MBR into the instruction register (IR).

ROM BITS:	PC:
K Poke instruction register	Jump INDI
M Clear IOMBR	
J PRIV	

(2) RAFE

RAFE is entered if the processor is in execute mode. The read address register (RAR) is used for the data fetch and 0 is put into the EXEQ flip-flop so that execute will be performed only once unless reinitiated by the instruction executed.

ROM BITS: PC:

P MA+1 is parameter Jump RAFF

K Poke MA

F Clear EXEQ flag

(17) RAFF

The RAFF step performs the actual memory cycle request for the execute mode data fetch. The memory cycle is awaited and the processor does not leave the step until the cycle has been granted. The left portion of the MBR is loaded into the instruction register (IR) and the incremented address is put back into the register.

ROM BITS: PC:

R Write Jump INDI

Register address = 0

P MA to bus

MA+1 is parameter

K Poke instruction register

M Request memory cycle

Wait on memory cycle

Poke MAP bits

Bit 4 data read or execute

(3) PCFE

PCFE is entered if the address contained in the program counter register (PC) is to be used for the instruction fetch, as is the case if the processor is in PROG mode.

ROM BITS: PC:

R Read, register address=2 Jump PCFF

P MA+1 is parameter

K Poke MA

(20) PCFF

PCFF behaves exactly as RAFF with the exception of the register address.

ROM BITS:	PC:
R Write	Jump INDI
Register address=2	
P MA to bus	
MA+1 is parameter	
K Poke instruction register	
M Request memory cycle	
Wait on memory cycle	
Poke MAP bits	
Bit 3 (program or stack read)	

(4) SPFE

SPFE is entered if the processor is in PEEL mode and the stack pointer register (SP) is to be used for the address.

ROM BITS:	PC:
R Read, register address=3	Jump SPFF
P MA+1 is parameter	
K Poke MA	

(21) SPFF

SPFF performs the memory access as in the PCFF and RAFF steps.

ROM BITS:	PC:
R Write	PC+1
Register address=3	
P MA to bus	
MA+1 is parameter	

K Poke instruction register

M Request memory cycle

Wait on memory cycle

Poke MAP bits

Bit 3 (program or stack read)

3.3.3 Instruction Decoding

At the top of sheet two the rectangle representing the INDI step is shown. This step decodes the group of the instruction and initiates dispatch 1. Although dispatch 1 is shown as two separate circles, it is, in fact, a single logical dispatch. The conditions of dispatch 1 are mutually exclusive. Instruction groups 0, 1, 4, 5, 6, and 7 dispatch to the load immediate sequence. Group 2 dispatches to a conditional load and group 3 dispatches to the multiple load and storage sequence.

(22) INDI

The INDI step decodes the instruction as just discussed.

ROM BITS:	PC:
R Write	Dispatch 1 (OPCI)
Register address=5	
P MBR right into register	

3.3.4 Load Immediate Sequence

Instruction groups 0, 1, 4, 5, 6, and 7 begin with a load immediate and are thus all dispatched to OPCI. In this step, the contents of the register to be loaded are loaded into

register 5, P2, of the processor. The register is then loaded with the new data by OPCA. If the instruction pushes a word onto the stack, three ensuing steps are initiated to write the word into memory and then the CHMO step is entered. The CHMO step, which is also entered directly from OPCA if the instruction does not push, allows the instruction to change the mode of the processor. After CHMO dispatch 3 is initiated and the processor either returns back to the MODE step (groups 0, 1) or goes to the display dispatches (groups 4-7).

(23) OPCI

OPCI is the beginning of the load immediate sequence. This step puts the contents of the register indicated by the instruction into P2.

ROM BITS:	PC:
R Write	Jump OPCA
Use instruction address for read	
Register address=5 (for write)	
P H1 or L0 reg to output	

(31) OPCA

OPCA loads the new value into the selected register unless inhibited by bit 13. The $\overline{@}$ symbol above the arrow in the OPCA rectangle indicates "not inhibit by bit 13," as bit 13 is the @ bit.

ROM BITS:	PC:
R Write unless @ inhibit	Jump CHMD unless
use instruction address for write	PUSH, then PC+1
P MBR right into register	

(32) OPCB

If the instruction indicates that a word is to be pushed onto the stack, OPCB will follow OPCA. OPCB puts the contents of the SP register (decremented by 1) into the memory address register (MAR).

ROM BITS:	PC:
R Read (not write)	PC+1
Register address=3	
K Poke MA	

(33) OPCC

OPCC puts the MAR onto the memory bus and writes the decremented stack pointer value into the SP register.

ROM BITS:	PC:
R Write	PC+1
Register address=3	
P MA to bus	

(34) OPCD

OPCD puts the contents of P2, which was loaded with the old value of the register being loaded by the instruction by the OPCI step, into the MBR right side. Part of the load immediate is copied into the left half of the MBR. OPCD also does the actual writing of the marked word into the stack. The clock pulse terminating this step is held until the memory responds to the write cycle request.

ROM BITS:	PC:
R Read (not write)	PC+1
Register address=5	

R HI or LO register to output

K Poke MBR

M Request memory cycle

Wait on memory cycle

Poke MAP bits

Bit 1 (stack write cycle)

(35) CHMD

This step allows mode changes if indicated by the instruction.

ROM BITS:	PC:
K Poke mode	Dispatch 3 (MODE)

(16) DOIT

The DOIT step is used for groups 4-7 to initiate the display dispatch sequence. DOIT loads the instruction register bits (IR) into the repeat status register (RSR).

ROM BITS:	PC:
K Poke RSR (IR to RSR)	Jump DIDI

(15) GRP1

GRP1 is a spare step which jumps control to mode.

3.3.5 Conditional Loads

The conditional loads of group 2 instructions have a special sequence of steps. COND holds until the pipeline is settled and no write operation is in process. This is necessary because the conditions to be tested by the conditional load are not settled until the pipeline is settled. When the pipeline has settled, the processor will move to CONA which may change the sense of the test. CONA then makes the test.

If the test is successful, control is transferred to CONB which does the loading. If the test was not successful, or after the loading has been performed, the processor goes back to the MODE state.

(26) COND

COND holds the processor to allow the pipeline to settle. If writing needs to be done, the WRITE subroutine is called.

ROM BITS:	PC:
none	Hold until SETTLED=DONE and call WRIT if WRITE REQ, then PC+1

(27) CONA

CONA makes the test on the selected condition. The sense of the test may first be reversed.

ROM BITS:	PC:
K Poke conditions	Jump MODE unless TEST COND, then PC+1

(28) CONB

CONB loads the register indicated in the instruction with the contents of P2, which was loaded from the right half of the MBR in the INDI step.

ROM BITS:	PC:
R Write Use instruction register address for write. Register address=5 (for read)	Jump MODE

3.3.6 The Multiple Load and Store Operation

The load and store instructions of group 3 are separated from the sink and retrieve instructions by dispatch 1. The load and store instructions go to the LOST step (for LOad and STore, not implying that the processor is lost.) Sink and retrieve instructions go to the step called SIRT. Both of these steps initiate dispatch 2. In the case of multiple load and store instruction, the step called GP3R is used following dispatch 0 to initiate dispatch 2.

Dispatch 2 is dispatched to one of two sequences. The Load and retrieve instructions simply require a data fetch by means of calling the DATA FETCH subroutine. Sink and store instructions are more complicated because they have to give instructions to the pipeline which will later result in the recording of the desired information in memory. Both sequences revert to the MODE step when completed.

(24) LOST

LOST loads the RAR with the contents of the right half of the MBR unless the inhibit bit (13) is high. The code for the read address register (RAR) is loaded into WHO, and the instruction word is loaded into the repeat status register (RSR).

ROM BITS:	PC:
R Write unless @ inhibit	Dispatch 2 (SIST)
Poke parameter	
Register address=0	
P MBR right into register	
MA+1 is parameter	
K Poke RSR (IR to RSR)	

(25) SIRT

SIRT performs the same task as LOST except that the data stack pointer (DSP) is used instead of the RAR.

ROM BITS:	PC:
R Write unless @ inhibit	Dispatch 2 (SIST)
Poke parameter	
Register address=6	
P MBR right into register	
K Poke RSR (IR to RSR)	

(5) GP3R

GP3R simply jumps from dispatch 0 to dispatch 2.

ROM BITS:	PC:
none	Dispatch 2 (SIST)

(40) LORT

LORT simply provides a suitable return address for the data fetch subroutine.

ROM BITS:	PC:
none	Call DATA then PC+1

(41) LORU

LORU counts the SHORT COUNT and jumps to MODE.

ROM BITS:	PC:
K SHORT COUNT	Jump MODE

(36) SIST

For sink and store operations the processor must wait until the pipeline is ready to accept new input data. SIST holds until the read buffer register flag (RBRF) is clear. If the pipeline is unable to accept new data, writing must be enabled to clear the pipeline, thus the WRITE subroutine is called.

ROM BITS:	PC:
none	Hold until *RBRF=DONE and call WRITE then PC+1

(37) SISD

When it is possible for the processor to give a new command to the pipeline, it does so in the SISD step.

ROM BITS:	PC:
K SHORT COUNT	Jump SISA
Poke command register	
M Raise outflag	

(73) SISA

SISA calls WRITE and waits for the pipeline to be cleared, that is, for all of the data requested to be written into memory. In multiple register transfers (where SC≠0) SISA does not wait because the next operation called for by MODE will be another GP3R transfer in the same direction as before. In such cases, only after the last transfer, does the processor actually wait in the SISA state.

ROM BITS:	PC:
none	Hold until *SC=0 or SETTLED= DONE and call WRIT if WRITE REQ, then PC+1

(74) SISB

SISB simply jumps to MODE.

3.3.7 Data Passing Operations

On the top of sheet 3 the data dispatches DIDI and DIDV are shown which separate the drawing instruction groups. Group 4 passes through a sequence which loads the RAR into WHO,

does a DATA FETCH, pokes the finite state machines controlling ABSOLUTE/RELATIVE and setpoint/endpoint functions. If the DO TWICE directive bit is set, the DIDU sequence for group 6 instructions is entered by the group 4 instruction, allowing a new command to be given with the same data.

The data-less instructions of group 6 are performed by going directly to DIDU and on to DINT where a command is sent to the pipeline and the processor outflag is raised. If the path is used by DO TWICE or group 4 instructions, the data passed will be swapped, which allows one to pack two 18-bit coordinates into a single PDP-10 word.

The DO INDIRECT instructions of group 5 pass through the sequence that starts with DIND. DIND initiates the memory sequence that fetches a pair of pointer which are loaded into P1 and P2 (registers 5 and 6) and used for two data fetches for the two endpoints of the line in question.

(6) DIDI

DIDI does nothing but separate out the group 5 instructions (DO INDIRECT).

ROM BITS:	PC:
none	Jump DIDV unless DO IND, then PC+1

(47) DIDV

DIDV counts the read count register and separates the DO DIRECT instructions of group 4 from the DO INTERNAL instructions of group 6.

ROM BITS:	PC:
K Count RCR	Jump DIDU if DO INT, else PC+1

(48) DDIR

DDIR loads the WHO register with RAR and does a data fetch by calling the DATA FETCH subroutine.

ROM BITS:	PC:
R Poke parameter Read (not write) Register address=0 MA+1 is parameter	call data

(51) DDIT

In the DDIT step, the finite state machines that control absolute-relative and setpoint-endpoint are incremented.

ROM BITS:	PC:
K Poke finite state machine	Jump MODE unless DO TWICE, then PC+1

(52) DIDU

If DO TWICE is indicated for a group 4 instruction, or if a group 6 instruction is indicated by DIDV, the DIDU step is entered. Here the processor waits for the input to the pipeline to be clear.

ROM BITS:	PC:
none	Hold until *RBRF=DONE and call WRIT if WRITE REQ, then PC+1

(53) DINT

DINT sends a command onto the pipeline and raises the processor output flag. If data are to be sent (group 4 instructions, DO TWICE) the data are swapped X for Y and Y for X.

ROM BITS:	PC:
K Poke Command register	Jump MODE
F Raise outflag	
M SWAP	

(7) DIND

The first step in the group 5 sequence is DIND, which initiates the memory cycle that fetches a pair of pointers.

ROM BITS:	PC:
R Register address=0 (RAR for read)	Jump DINE
P MA+1 is parameter	
K Poke MA	

(42) DINE

DINE makes the actual memory cycle requests and waits for the request to be granted. In addition, DINE loads the RAR with the incremented value from the MAR and counts the read count register (RCR)

ROM BITS:	PC:
R Write	PC+1
Register address=0	
P MA to bus	
MA+1 is parameter	
K Count RCR	
M Request memory cycle	
Wait memory cycle	
Poke MAP bits	
Bit 4 (data read or execute)	

(43) DINF

DINF loads the memory buffer register left side (MBR) into P1 and puts "P1" in the WHO register.

ROM BITS:	PC:
R Write	PC+1
Poke parameter	
Register address=4	
P MBR left into register	
MA+1 is parameter	

(44) DING

The DING step loads the right half of the MBR into P2 and calls a data fetch.

ROM BITS:	PC:
R Write	Call DATA
Register address=5	
P MBR right into register	

(45) DINH

DINH pokes the finite state machines and puts P2 in WHO and calls a data fetch.

ROM BITS:	PC:
R Poke parameter	call DATA
Register address=5 (read)	
P MA+1 is parameter	
K Poke finite state machine	

(46) DINI

DINI again pokes the finite state machine and then returns

control to MODE.

ROM BITS:	PC:
K Poke finite state machine	Jump MODE

3.3.8 WRITE and DATA FETCH Microcode Subroutines

These two subroutines are used by the other microcode steps. The DATA subroutine also uses the WRITE. Because the microcode can only store a single level of subroutine return, the fact that WRITE is being used by DATA is stored in flip-flops called F1 and F2. These are actually the two sense flip-flops on the Microcode Control card.

The WRITE subroutine begins with the WRIT step which initiates dispatch 4. Dispatch 4 dispatches to one of 3 states as determined by kind of write to be performed. The 3 write sequences differ only in the register used for the memory address. The register address is loaded into the memory address register (MAR), and a memory cycle is requested and awaited. All three sequences exit to the WRIA state and from there either back to the step which called the subroutine or back to the DATA FETCH subroutine.

(54) WRIT

WRIT simply initiates dispatch 4.

ROM BITS:	PC:
none	Dispatch 4 (DSPW)

(55) DSPW

DSPW is entered if sink instructions have called for the write. The contents of the data stack pointer (DSP) are used

as the memory address.

ROM BITS:	PC:
R Register address=6 (read)	PC+1
P MA+1 is parameter	
K Poke MA	

(56) DSPA

DSPA makes the memory cycle request and does not exit until the request has been granted. MA+1 is loaded back into the DSP register.

ROM BITS:	PC:
R Write	Jump WRIA
Register address=6	
P MA to bus	
MA+1 is parameter	
M Request memory cycle	
Wait on memory cycle	
Poke MAP bits	
Bit 2 (data write cycle)	

(57) RARW

If a store instruction calls the write, the RARW state is entered and the contents of the read address register (RAR) are used for the memory address.

ROM BITS:	PC:
R Register address=0 (read)	PC+1
P MA+1 is parameter	
K Poke MA	

(60) RARA

(60) RARA

RARA makes the memory cycle request exactly as DSPA. MA+1 is loaded back into the RAR.

ROM BITS:	PC:
R Write	Jump WRIA
Register address=0	
P MA+1 is parameter	
MA to bus	
M Request memory cycle	
Wait on memory cycle	
Poke MAP bits	
Bit 2 (data write cycle)	

(61) WARW

If the processor is in memory to memory mode, data will come out of the pipeline in response to ordinary instructions of groups 4, 5, 6, and 7. Such data is written into memory using the write address register (WAR) as an address. These cases enter the WARW state. WARW loads the WAR into the MAR and counts the write count register (WCR).

ROM BITS:	PC:
R Register address=1 (read)	PC+1
P MA+1 is parameter	
K Count WCR	
Poke MA	

(62) WARA

WARA makes the memory cycle request and reloads the WAR with MA+1.

ROM BITS:	PC:
R Write	PC+1
Register address=1	
P MA to bus	
MA+1 is parameter	
M Request memory cycle	
Wait on memory cycle	
Poke MAP bits	
Bit 2 (data write cycle)	

(63) WIRA

WIRA acknowledges the write buffer request flag and exits to the step calling the subroutine if F1 is clear and to WIRB if F1 is set.

ROM BITS:	PC:
M Acknowledge write buffer request flag	Exit to state +0 if F1 clear, else PC+1

(64) WRIB

WIRB does nothing but check the state of F2 to determine whether to return control to DATA or DATD.

ROM BITS:	PC:
None	Jump DATD if F2 is set, else PC+1 (DATA)

The DATA FETCH subroutine has the job of fetching one (2D) or two (3D) words of data from memory and raising the output flag. The DATA FETCH subroutine not only fetches a set of coordinate data, but also initiates the pipeline action on it. The data sequence is repeated for 3D transfers by entering the DATC, DATD, DATE sequence which perform exactly the same operations as the corresponding DATF, DATA, and DATB steps of the

subroutine.

Before each data fetch, the processor must be sure that the pipeline can, in fact, accept the new data. The DATA and DATD steps thus have a hangup on RBRF (read buffer register flag) and call the WRITE subroutine to clear the pipe line. DATA sets F1, but clears F2, while DATD sets both F1 and F2, thus allowing the WRITE subroutine to determine the state to which it should return control.

The steps DATB and DATE initiate the data fetch operations. Here the register named by WHO is passed to the MAR and possibly decremented by 1. The decrement is controlled by the fourth bit of the WHO register and applies only to the DSP. DATB separates the 2D from the 3D case to omit the second data fetch for 2D.

DATC and DATF call the read cycle. The clock pulse for these steps is delayed until the memory cycle is complete and the data called for are in the memory buffer register. These steps also transfer data from the memory buffer register (MBR) to the read buffer register (RBR) and raise the processor output flag to initiate the pipeline processes. The DATA FETCH subroutine exits to the instruction following the one that called it as indicated by the pigtail marked EXIT +1.

(65) DATA

The DATA step sets F1 and clears F2. The state is not left until the pipeline is clear as indicated by *RBRF and *WRITE.

ROM BITS:	PC:
F Set F1	Hold until *RBRF = DONE and call WRITE if WRIT REQ
Clear F2	then PC+1

(66) DATB

In the DATB state, WHO specifies a register whose contents are loaded into the MAR. If the 4th WHO bit indicates a decrement and the WHO register name is DSP, the address is decremented. DATB separates the 2D and 3D cases. In 2D, the exit is to DATF to skip the second data fetch.

ROM BITS:	PC:
R Use parameter register for read	Jump DATF unless 3D, then PC+1
K Poke command register	
Poke MA	

(67) DATC

DATC loads the contents of the MAR, possibly incremented, into the register indicated by WHO. A data read request is made and the contents of the MBR are loaded into the RBR. The outflag is raised.

ROM BITS:	PC:
R Write	PC+1
Use parameter register for write	
P MA to bus	
K Poke RBR	
F Raise outflag	
M Request memory cycle	
Wait on memory cycle	
Poke MAP bits	
Bit 4 (data read or execute)	

(70) DATD

DATD behaves exactly like DATA except that both F1 and F2

are set.

ROM BITS:	PC:
F Set F1	Hold until *RBRF=DONE and call WRIT if WRITE REQ, then PC+1
Set F2	

(71) DATE

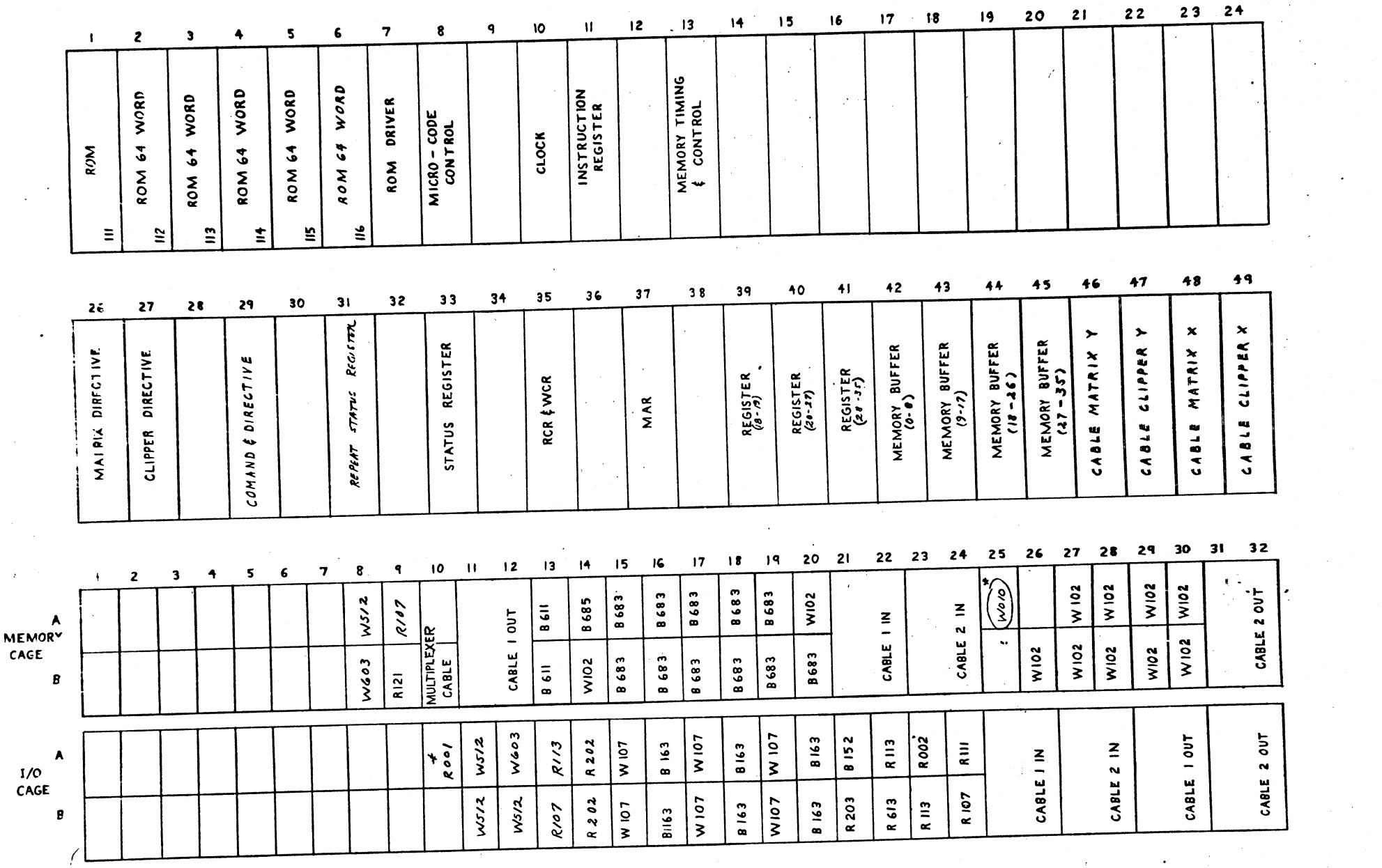
DATE is identical to DATB except that the command is not poked.

ROM BITS:	PC:
R Use parameter register for read	PC+1
K Poke MA	

(72) DATF

DATF performs the functions of DATC and in addition, clears F1 and F2. DATF exits to the state after the state calling the subroutine.

ROM BITS:	PC:
R Write	Exit to state calling Subroutine +1
Use parameter register for write	
P MA to bus	
K Poke RBR	
F Rasie outflag	
Clear F1	
Clear F2	
M Request memory cycle	
Wait on memory cycle	
Poke MAP bits	
Bit 3 (program or stack read)	



EVANS & SUTHERLAND COMPUTER CORP.
DISPLAY PROCESSOR
CHANNEL CONTROL - PDP10 INTERFACE
SHEET 2 OF 14
DATE 4-7-69
101102-900
Richmond Hurlbut 2/28/69

Figure 4.1

PROCESSOR BACK PANEL

4.1 Introduction

The Channel Control Logic is contained in two card cages. The upper cage contains the control logic for the processor, while the lower cage contains the memory buffer and the registers of the processor. Below the processor cages are two cages which hold the DEC interface logic. The upper of these contains the memory interface and the lower contains the I/O interface.

4.2 Reference

E & S block diagram No. 101102-900, Channel Control - PDP-10 Interface.

4.3 Card Placement and Interconnections

We will first deal with the placement of the processor cards in their cages. Included in this discussion will be very general references to the function of each of the cards. Intercard connections will then be examined.

4.3.1 Card Placement

Figure 4.1 constitutes a stuffing chart for the processor and interface cages. The cards in the upper cage provide most of the necessary control logic for the processor. The cards in this cage are listed below.

Card name and number:	Quantity:	Function:
Instruction	1	Captures and partially decodes the instruction bits sent to the processor by the PDP-10.
Microcode Control	1	Contains the test logic and program counter.
ROM Driver	1	Decodes the ROM address developed by the program counter and drives the selected ROM word.
ROM 64 Word	6	Provides the ROM control bits.
Memory Timing and Control	1	Provides control and synchronization for the interface communications.
Clock	1	Generates the clock pulse and controls lamp driver.

In addition to these cards, two small Lamp Driver boards are mounted on the outside of the upper cage. These boards control the panel lights and interface between the control panel switches and the logic of the processor.

The lower processor cage contains the memory buffer, the registers of the processor, and the directive and data cables which allow communication between the processor and the other devices in the LDS system. The cards and cables in this cage are listed below.

Card name and number:	Quantity:	Function:
Memory Buffer Register	4	Contains the memory buffer register and associated read buffer register and write buffer register.

Register	3	Contains the 8 lower numbered registers of the processor.
Memory Address Register	1	Registers and sends to computer memory the memory address selected by the processor.
Counter Registers	1	Contains two counting registers that count the number of memory read and memory write operations executed.
Status Register	1	Contains a register used to store conditions to be tested by conditional instructions.
Repeat Status Register	1	Contains a register used to store the instruction to be performed in repeat mode and associated instruction update logic.
Command and Directive	1	Contains the directive register and the low 16 bits of the command register.
Clipper Directive Cable	1	Cables containing the directive bits for the Clipping Divider and Matrix Multiplier Data I/O cables for the Clipper and Matrix Multiplier.
Matrix Multiplier Directive Cable	1	
Cable Matrix X	1	Data I/O cables for the Clipper and Matrix Multiplier
Cable Matrix Y	1	
Cable Clipper X	1	
Cable Clipper Y	1	

4.3.2 Memory Buffer Input

The Channel Control not only communicates directly with the PDP-10, but also controls the communications between the PDP-10 and the other devices in the LDS system. These communications are effected through the memory buffer and controlled by the

memory timing and control logic of the processor.

There are four sets of inputs for the memory buffer register. Input can come directly from the memory interface, from the I/O interface, from external devices via the write buffer register (WBR), and finally, from the processor itself.

The MBR inputs of the Memory Buffer Register cards are wired directly from the PDP-10 memory interface cable. The data on these lines is enabled into the register when the *LOAD MBR signal, which is wired from the Memory Timing and Control card (MTC), goes low.

The IOBB input bus of the MBR cards is wired from the PDP-10 I/O interface cable. This data is registered when the *LOAD IOB signal from the MTC card goes low.

The WBRB inputs of the MBR cards are wired from the clipper and matrix multiplier data cables. This data is first registered in the Write Buffer Register and sent from there to the memory buffer. The *LOAD WBR signal that enables this operation comes from the MTC card.

In order to allow push word operations, 8 bits from the instruction register and 18 bits from the RMIB bus, which can contain the contents of any of the 15 processor registers, can be loaded into the MBR through the PUSHW inputs and then written into memory. The high nine bits of the PUSHW inputs are wired to ground to form a load immediate without a push instruction. Bit 13 is also wired low to insure that the load will take place. The 4 IR bits indicating the register to be loaded are wired to bits 9-12. The mode flip flops are coded to set the mode correctly and wired to PUSHW bits 14-17.

The RMIB bus originates at the OUTA, OUTB bus of the Register cards and is wired to the PUSHW inputs of the MBR cards for bits 18-35. The *LOAD PUSHW signal for enabling this input also comes from the MTC card.

4.3.3 Memory Buffer Output

The contents of the memory buffer may be outputed directly to memory if a memory write operation is being executed. The MWB output bus of the MBR card is wired to the memory interface for this purpose.

The data in the MBR can also be transferred to an external device if a memory read operation has just been executed. The data is first loaded into the read buffer register (RBR). The LOAD RBR signal that enables this loading is a ROM signal. The RBR output bus is wired to the data cables for the clipper and matrix multiplier. The bits 0-17 are wired to the X coordinate cables, and the bits 18-35 are wired to the Y coordinate cables.

The left half of the MBR (bits 0-17) may also be loaded into the instruction register. These bits are wired from the MBR output bus to the corresponding bits on the MBR input bus on the Instruction card.

And finally, either the right or left half of the MBR may be loaded into any of the registers of the processor. The wiring involved is discussed in section 4.3.8.

4.3.4 Instruction and Control Logic

The data from the left half of the MBR is loaded into the

instruction register for use in controlling the operation of the processor when the ROM signal PK11, which is wired to POKE IR, goes high. The 18 bits of instruction register output (IR) are wired from the instruction card to the repeat status register. The Repeat Status Register card generates the command (COM) bits 0-13 and the SWAP bit, which are wired to the Directive cable.

Instruction register bits 14-17, as well as *13, are also wired from the Instruction card to the Status Register card. These bits have to do with register address and are used along with the CODE A outputs generated by the Instruction card to determine address selection.

The CODE A signals are controlled by ROM signals which dictate which of the possible register addresses to use. The ROM signals REC4 and REC5 are wired to the read selection signals UPFR (use parameter for read) and UIFR (use instruction for read) respectively. Selection of the register to be used in writing is determined by the ROM signals REC6, REC7 and REC8. These signals select U5FW (use register 5 for write), UPFW (use parameter for write), or UIFW (use instruction for write) respectively. In both read and write selection, if none of the possibilities is chosen, the ROM signals feeding GRA(1), GRA(2), and GRA(3) are used to specify the address. These signals are also used to load the parameter register. This parameter register (WHO) is also located on the Instruction card and indicates register addresses to be used and whether to count the memory address up or down on the MAR card. WHO is loaded from the GRA

inputs when the ROM signal REC3 which is wired to POKE PARAMETER goes high. The up/down portion of the parameter is fed by a ROM signal tied to MARUP. The output developed, MARUPC, is wired to MA + 1 → R on the MAR card.

CODE A outputs from the Instruction card are wired to the CODE A inputs of the Register card as well as to the CODE A inputs of the Status Register card. The write selection signals developed by the SR card from the CODE A inputs are enabled by ROM signals REC1 and REC2, which are wired to WR and WRIFOK. It should be noted that the CODE A wiring is not consistantly bit by bit. Figure 4.2 below shows the actual wiring of the CODE A bits.

ADDRESS CODE WIRING

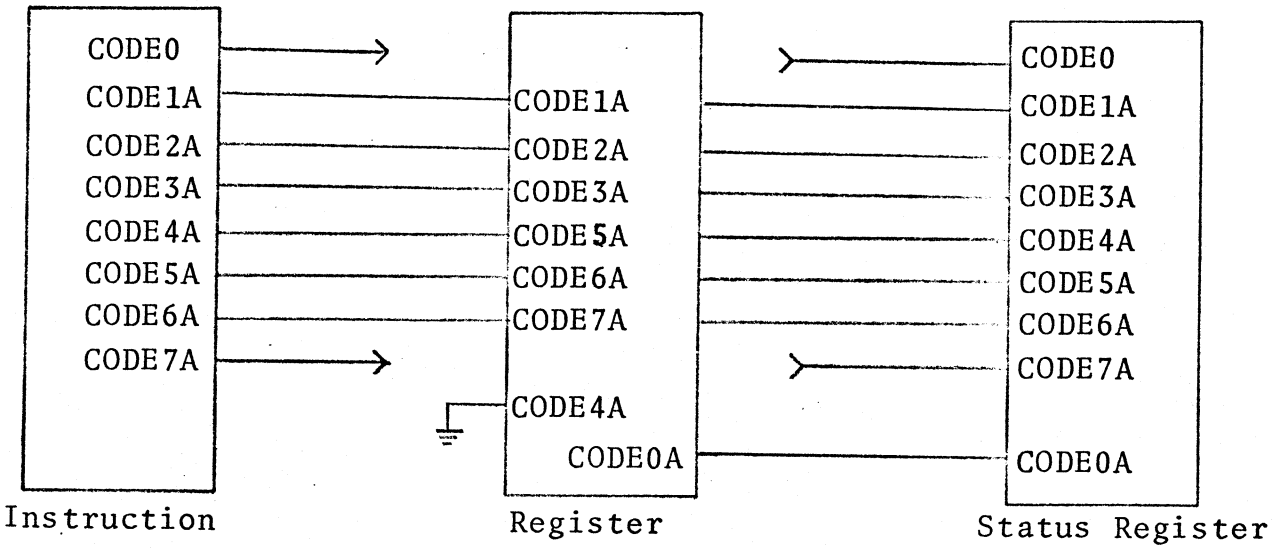


Figure 4.2

The mode flip flops are also located on the Instruction card and get loaded when CHANGE MODE goes high. CHANGE MODE is wired to the ROM signal PKC3. Two inverters on the Instruction card invert the ROM signals PAC2 and PAC3, which are sent to

the Register cards to select ENOA, ENOB or EOA.

The Instruction card also develops the small binary offset numbers needed by the microcode for dispatching. The *EXAD bits 6, 7, and 8 are wired from the Instruction card to the Microcode Control card. The remainder of the *EXAD bits on the Microcode Control card are tied to Vcc. Selection of the dispatch is controlled by ROM signals TSC4-6 which are wired to the SNA, SNB, SNC on the Instruction card.

The D (done) signal needed by the microcode is also generated by the Instruction card logic and wired from the DONE output.

ROM signals TSC0-3 are used to select one of the following DONE CONDITIONS:

TSC0	SC = 0 (short count = 0)
TSC1	SETTLED (pipeline clear)
TSC2	PAUSE REQUEST
TSC3	RBRF (read buffer register full)

The Microcode Control card contains test logic to test certain conditions within the processor that affect the program counter update. The signals tested and the cards where they originate are listed below.

200	not used	not used	not used	DO TWICE (C+D)
100	STOP (SR)	WRITE REQ (MTC)	PAUSE REQ (SR)	IOMBR (MTC)
40	DO IND (RSR)	DO DIR (RSR)	DO INT (RSR)	PUSH (IR)
20	@ (IR)	3D (C+D)	SC = 0 (RSR)	COND OK (IR)
	10	4	2	1

COND OK indicates that the test selected onto TEST BUSS, an open collector bus wired from the Status Register and Counter Register cards to the Instruction card, is satisfied. IR bits 4 and 5 are sent to the Status Register and Counter Register cards to J or K the test conditions. IR bits 14-17 are decoded on the Status Register card and are used to select one of the possible test conditions. All test bits, except those dealing with the sign bits of the counter registers, are located on the Status Register card itself. Thus, only TEST 10 and TEST 11 are brought off the SR card. These signals are wired to corresponding inputs on the Counter Register card.

The MEM TO MEM signal on the SR card, which is wired from DIR (34) of the Command and Directive card, is ANDed with WCR18 from the Counter Register card to force a STOP condition. STOP ON HIT is wired from DIR(35) on the C+D card to the SR card where it is ANDed with the HIT bit from the Clipper directive to produce a STOP.

ROM signals JUM 2-7 are used to drive the YES, NO, and SEL signals on the Microcode Control card. ADDR(1) and ADDR(2) are wired to ground since the address field is only 6 bits. ROM signals JUA3-8 are used to drive the low 6 bits of ADDR. JUM1 drives PUSHR which is used for subroutining. ROM signals +TSC7-9 and TS10-14 drive the CI signals which are used to select test conditions. The two flip flops on the Microcode Control card are used to mark exits from the WRITE subroutine back into the DATA FETCH subroutine. ROM bits FCC5-8 allow setting and clearing of these flip flops.

The ROMAS outputs generated by the program counter on the Microcode Control card are wired to the respective ROM Driver and Clock card inputs. The ROM Driver uses this address to select and drive one of the ROM words. The ROM cards are wired in parallel from the ROM Driver outputs.

4.3.5 Memory Timing and Control

The output of the parity net PARITY (1-9) for each MBR card is wired into the appropriate *OPAR input of the Memory Timing and Control (MTC) card. These signals contribute to the parity checking and generating network.

The *SETMBR1 and *SETMBR2 signals used to set the memory buffer register before loading the data on PUSHW, WBR, or IOB inputs are wired from the MTC card, as is the CLRMBR signal needed to clear the MBR before loading from the memory bus. As already noted, all of the load enabling signals (*LOADMBR, *LOADPUSHW, *LOADIOB, and *LOADWBR) are also generated on the MTC card and passed to the corresponding inputs on the MBR cards.

Two directive bits from external devices, WRITE RAR and WRITE WAR are captured on the MTC card and then sent to the Instruction card to be used in dispatch control.

The Memory Timing and Control card behaves as if it were an external device to the processor. When the processor needs a memory cycle, it raises a device flag which the MTC card watches (CYCLE FLAG). After the memory cycle is successfully completed, the MTC card sends an acknowledge signal (ACK) back

to the processor clock card.

The kind of write to be executed (DATA or STACK) is determined by two ROM bits, IOC8 and IOC9. PKC4 from the ROM indicates that the MBR is to be written from the PUSH bus (PUSH TO MBR). The ROM signal IOC3 is raised at the end of the write subroutine to give WBRACK to the MTC card.

CLRIOMBR on the Memory Timing and Control card is driven by FLC9 and is used to clear the IOMBR flip flop after the DATAO instruction has been started. The ROM signal IOC2 indicates to the MTC that the processor is paused. This allows a DATAO SET from the I/O bus interface to set the IOMBR flip flop.

Upon receiving a write request from an external device such as the clipper, the MTC accepts the request at some time and returns a *DATA (*DATA B in the case of the clipper) to the device telling it to put its data on the WBRB lines.

DATA READY TO C from the Clock card is the input flag to the clipper. If it were not for the fact that the flag should be raised one clock cycle after the ROM has requested it, FLAG B would be considered the clipper flag. But, in order to cause this delay, FLAG B is wired to RAISE output FLAG C, thus making FLAG C the clipper flag. Because both flags are raised, both must be acknowledged, so the acknowledge signal from the clipper is wired to both ACK FROM B and ACK FROM C on the Clock card. Because FLAG B is raised first, it is sent to the Instruction card (RBRF) to determine done conditions.

FLAG A is used to request a memory cycle and is raised by

the ROM signal IOC4 which is wired to RAISE OUTPUT FLAG A on the Clock card. The OUTPUT WAIT A signal needed by the Clock card is supplied by ROM bit IOC5.

LONG CYCLE on the Clock is wired to ground to assure the shortest possible clock cycles for the clock switch setting. Unlike the clipper, the processor clock is distributed directly to the system from the Clock card. No buffering is employed. Clock (1) drives the MBR cards; Clock (2) drives CPA of all the Register cards; Clock (3) drives the clock signals of the Repeat Status Register, Command and Directive, Microcode Control, and Instruction cards; Clock (4) drives the MAR, Counter Register, Status Register, and Memory Timing and Control cards.

4.3.6 Register Input

There are 15 basic registers of the processor. (Since the MAR has the ability of being loaded immediately or pushed onto a stack, it is also considered as a basic register). These 15 registers fall into two groups. The first group, registers 0-7, are contained on the Register card. Those registers not on the Register card and numbered 8-15 constitute the second group. All of the input and all of the output (with the exception of MAR output direct to memory) for both groups of registers pass through the Register card.

There are 4 possible sources of register input. Input can come from the left half of the MBR, from the right half of the MBR, from the SB bus, which is an open collector output bus common to all of the higher numbered registers, and finally, from the output of the lower numbered registers through an

internal connection on the Register card.

The left half of the MBR comes to the NOA, NOB inputs on the Register card. Because the register input selection switch inverts the signals, the compliment MBR outputs (*MBR) are wired from the Memory Buffer Register card.

The right half of the MBR is wired to the INA, INB inputs of the Register card. Data coming into the Register card on this input bus passes through two stages of inversion before it is registered, so the true MBR inputs (bits 18-35) are taken.

The contents of any of the registers not on the register card can be read onto a common open collector bus (SB) when the appropriate *READ signal is driven low. These READ signals are wired from the Status Register card to the card containing the register to be read. This SB output bus is wired to the POA, POB input bus of the Register card. The data on SB is in compliment form and thus registered in true form after the one stage of inversion contained in the POA, POB lines.

The input for the registers not on the register comes from the OUTA, OUTB outputs of the Register card. This bus is labeled RMIB, and is wired to the RMIB input busses on each of the cards containing registers.

The ROM signal PAC4, which is wired to REG → REG on the Status Register card is used along with other logic to derive the signals *LOREG → REG and *HIREG → OUTPUT which are wired to Register card inputs *ERA, *ERB and *EPOA, *EPOB respectively.

To load the Memory Address register, the POKEMA signal

on the MAR card must be high. This signal is wired from PK12 of the ROM. The MA → REG signal on the Status Register is also a control signal for register loading. MA → REG is wired from the ROM signal PAC6.

4.3.7 Register Output

The contents of the lower numbered registers can be read onto the RA, RB output bus which is tied to the RA, RB input bus on the Memory Address Register card. Data from the other registers can be outputed to the registers on the Register card via the SB bus.

The directive register (DIR) feeds the lower 16 bits of the Command register. The high 14 bits of the Command register and the SWAP bit are wired from the RSR bits. The Command register is loaded when POKE COM, which is wired from the ROM signal PKC5 to the Directive and RSR cards. The COM output bits are wired to the directive cable.

4.3.8 Directive Cable Bits

The directive cable carries directives to and from the Clipping Divider. The bits of this cable, some of which have already been discussed, are listed below.

PIN NUMBER: (only odd)	TO/FROM:	SIGNAL NAME:	CARD FROM WHICH SIGNAL ORIGINATES OR TERMINATES:
1 - 27	TO	COM(0-13)	Repeat Status Register
29 - 35	TO		Vcc
37 - 67	TO	COM(18-32)	Command & Directive
97	FROM	CLIPCL	Status Register

PIN NUMBER: (only odd)	TO/FROM:	SIGNAL NAME:	CARD FROM WHICH SIGNAL ORIGINATES OR TERMINATES:
71	FROM	CLRHIT	Status Register
73	FROM	SETHIT	Status Register
75	FROM	CLRAIC	Status Register
77	FROM	SETAIC	Status Register
87	FROM	TUT TUT FORBID	Status Register
69	FROM	CLIPPER SETREG	Status Register
81	TO	MC 10	Memory Timing & Control
83	FROM	ACK B	Memory Timing & Control
85	FROM	REQ B	Memory Timing & Control
89	FROM	WD(22)	Memory Timing & Control
91	FROM	WD(21)	Memory Timing & Control
93	FROM	ACK FROM B	Clock
95	TO	DATA READY TO C	Clock

4.3.9 PDP-10 Interface Wiring

Most of the wiring from the processor to the memory interface is wired with twisted pair to eliminate asynchronous signal changes inducing noise into the processor.

The output of the memory bus receiver (W102) labeled MRBi on the PDP-10 memory interface print are wired to the level converter inputs MRB(j) on the MRB card (pullup resistors are located at the level converter inputs). The MWB(j) level converter outputs on the MRB card are wired to the inputs, MWBi, on the memory bus transmitters (W102).

The output of the I/O bus receivers (W107), IOBiB are

wired to the level converter inputs IOBB(j) of the MRB card.

All of the above-mentioned signals are level converted on the MBR card. They are transmitted between the processor and interface at DEC logic levels (ground and -3 volts). All other signals except the MA signals are passed to or from the interface at TTL logic levels and are converted in the interface by W603 or W512 DEC modules.

RESUME and PAUSE REQ and CONO signals sent from the I/O bus through level converters to the Status Register card where they are stored in flip flops at the occurence of the signal CONO SGTB.

The program flags, HIT bit, and AIC bit are sent from the Status Register card to the I/O interface to be picked up by a CONI instruction. The PROGRAM STOP and TUT TUT FORBID levels are brought over in a similar manner.

The ROM signals IOC1 and IOC2 come over to the I/O interface to indicate STOPPED or PAUSED for the CONI instruction.

The high-order bit of both Count Registers is brought across for collection in the CONI.

MC10 is sent to the MTC card where it triggers a one shot to provide correct length for the processor.

The PARITY ALARM and NXM levels are sent across to go into the CONI and also to cause an interrupt if selected.

DATAO SETB and DATAO CLRB are passed to the MTC card to clear and set the IOMBR flip flop.

The following signals are wired from the MTC card to the memory interface for communication:

IREQ	-	internal request	from interface
*SIREQ	-	set internal request	to interface
READ	-	read cycle	to interface
WRITE	-	write cycle	to interface
*RORSG	-	read restart	from interface
WRITM	-	write restart	to interface
IGNPAR	-	ignore parity	from interface
MRBP	-	memory read parity	from interface
MWBP	-	memory write parity	to interface

The memory address bits from the MAR card are wired to the inputs of the B683 or to inverters to send negated MA bits over the bus. The inverters are shown on the memory interface prints. These signals are level converted on the MAR card and passed at DEC logic levels. The prints show the wiring as it should appear for connection to the AMPEX memory. The wiring is actually as DEC memory requires. The changes made are as follows:

MA17(0) is wired to MA22(0) by means of an inverter.

MA17(1) is wired to -3v to provide *FMC SELECT.

MA16(1) is wired to ground to provide FMC SELECT.

MA34(0) is wired to MA35(1) and the inverter is disconnected.

MA16(0) is wired to MA21(1).

These changes must be restored before connecting to the AMPEX memory.

MEMORY BUFFER REGISTER

5.1 Objective

The Memory Buffer Register card provides the necessary buffering for communications between PDP-10 memory, the Channel Control, and external devices. A write buffer register is provided for a first line of buffering of information coming from an external device that is to be written into PDP-10 memory. A memory buffer register communicates directly with PDP-10 memory after the necessary level changes have been made. Data coming into the memory buffer register that is to be sent to an external device is loaded into a read buffer register which provides a line of buffering before this information is passed on to the external devices in the system.

5.2 Reference

E & S logic diagram No. 101114-600, Memory Buffer Register.

5.3 Card Contents and Function

If data from an external device within the system is to be written into memory, it is brought into the write buffer register (WBR). Data from the WBR is loaded into the memory buffer register (MBR) and from there the data is transferred to PDP-10 memory. If an external device requests a memory read, data is sent from memory to the MBR and then transferred from the MBR into the read buffer register (RBR). The output of

the RBR supplies the information to the device making the request for the purposes of parity checking. A parity net is incorporated into the Memory Buffer Register card.

In the following discussion, the write buffer register operations will be described first. Descriptions of the operations of the read buffer register and the memory buffer register will follow. A final note on the parity net will also be included.

5.3.1 Write Buffer Register

The WBR can receive input data from several different devices in the system. Input comes via an open collector bus which feeds the WBR inputs on the Memory Buffer Register card. The open collector gates for the bus are usually found on the card on which the cable from the device in question terminates. However, the necessary pull up resistors are on the Memory Buffer card itself. The data is registered in a D-type latch when strobed by the signal "a" which is driven by *LOAD WBR from the Memory Timing and Control card. The output of the latch is sent as input to the MBR.

5.3.2 Read Buffer Register

Information to be sent to external devices is sent via the RBR. The RBR is comprised of D-type latches fed by the output of the MBR. Clocking is controlled by a logical AND of the LOAD RBR and the processor clock pulse. The compliment

output of the register latches are taken and inverted by a power driver which gives the signal sufficient drive to drive several devices.

5.3.3 Memory Buffer Register

The MBR has several possible inputs. Data may be coming from the processor itself, in which case it will come in on the PUSHW lines. Alternatively, data can come from the PDP-10 I/O bus, from the PDP-10 memory bus, or from the write buffer register. The register is made up of flip-flops comprised of a 2-input low-level NOR gate and a 4-wide AND-OR-Invert gate. The flip-flop is capable of operating in a clear-set mode, which is used in registering memory input data (MRIB), or in a set-clear mode used for loading the other inputs into the register.

In the following discussion, the bug positions and pin numbers referenced can be found on sheet one of the logic drawing. Bugs 41 and 51 act as two NAND gates wired as a latch. Under normal operations, signals "e" "f" "g" are held low by high signals on *LOAD PUSHW, *LOAD IOB, and *LOAD MBR respectively. Signals "c" and "d" are held high by a low signal on CLR MBR and a high signal on *SET MBR.

The level converters shown in positions 74 and 76 convert the DEC levels to corresponding TTL levels. The memory bus from the PDP-10 is normally at -3V (i.e., unless a 1 is being written into the MBR). This is converted by the level converter to a +5V and the ground level is passed as ground.

Thus, the input at pin 4 of bug 51 is high unless a 1 is being sent from memory. Assuming these normal states as described, we can now examine the operations of the flip-flop.

The transfer of memory data coming in on the MRIB bus follows the clear-set sequence. First, "c" is pulsed low, which clears the flip-flop prior to the memory read pulse from the PDP-10. If a logical 1 is sent from memory, pin 4 of bug 51 will drop low which sets the MBR flip-flop. If a logical 0 is sent, the flip-flop will, of course, remain in the 0 state.

For transfers of data from the WBR, the PDP-10 I/O bus, or the PUSHW input, the set-clear sequence is used. The point "d" is pulsed high which sets the MBR flip-flop. One of the selection signals "e" "f" or "g" will be driven high selecting the desired input. A high signal on these inputs will then clear the flip-flop. The MBR thus contains the compliment of the data present on pins 9, 3, and 13 of bus 51.

5.3.4 Parity Net

A parity net is included on the Memory Buffer Register card to allow parity checks. Output of the MBR flip-flops is fed into this parity net. The PARITY (1) signal, which is high when an even number of 1's are in the register, is sent to the Memory Timing and Control card for the parity check.

MEMORY ADDRESS REGISTER

6.1 Objective

The Memory Address Register card stores the memory address to be used in memory access operations. Adders located on the card allow the addition of -1 to the address before accessing, or alternately, the addition of +1 after the address has been used for the access. Electrical interfacing with the computer is allowed by the level converters contained on the card.

6.2 Reference

E & S logic drawing No. 101134-600, Memory Address Register.

E & S block diagram No. 101101-900, PDP-10 Display Processor.

6.3 Card Contents and Function

The Memory Address Register consists of an 18-bit down counter, an 18-bit register, and 18-bit up counter, level converters and logic to control selection and carry prediction. Data comes first into the down counter. This data, which is normally in compliment form, may be decremented if appropriately enabled. The output of the down counters is registered when the clock pulse is enabled. The compliment output of the registers goes to the level converters and to the up counters which add +1 if enabled. The output of the up counters forms the RB output bus which is sent back to the internal registers of the

processor. The following discussion will follow the data path as described.

6.3.1 Down Counter

The down counter is composed of five four-bit adders with one set of inputs grounded. (The two high-order bits are unused.) Data from the address registers of the processor comes into these adders on the RA input bus. If the data is to be incremented, the signal MA+1 TO R drops low and a carry in sequence is initiated. A carry prediction chain is provided which ANDs the signals on the RA bus until a zero state is encountered, which then inhibits a carry in for the next adders. Adder output is then sent to the address register.

6.3.2 Address Register

The memory address register consists of D-type latches. Strobing is controlled by the POKE MA signal which is ANDed with the clock pulse. As the data on the RA input bus is normally in compliment form, the data stored in the registers must also be considered as compliment information. For this reason, the compliment output of the register is taken, thus putting the data in true form. This true data is sent to the level converters for memory access and to the up counter.

6.3.3 Level Converters

Each level converter is controlled by a 2-way switch (2-wide AND-OR-Invert gate) that chooses one of two possible inputs.

If MA TO MADR is high, the switches select the output of the memory address register for both halves of the address. Address data from external sources may also be taken. Input to the other side of the selection switches comes in on the EXT bus for the lower 9 bits of address. The signal EXT TO MADR enables the selection of this data. The MAP input bus feeds the upper 9 bits of address and is selected by the MAP TO MADR signal.

The selected signal is converted by a level converter comprised of discrete components which inverts the signal (which has been inverted by the selection switches) and makes the necessary level change. Ground passes as ground, and +5V is converted to -3V.

6.3.4 Up Counter

The output of the memory address register is also sent to the up counter. This counter is also composed of 4-bit adders with one set of inputs grounded. As the data is now in true form, an up count is effected by initiating a carry. The carry in/carry out sequence of the adders is used to perform the carry. A high signal on MA+1 TO MBR is required to initiate this carry. Thus, the signal initiating a carry for the up counter is the compliment of the signal that initiates the carry for the down counter.

Adder output is sent onto the open collector bus RB via open collector NAND gates which are enabled by the signal MA→RB. This allows the new address to be recycled into the internal storage registers of the processor.

REGISTER CARD

7.1 Objective

Register cards are used to make up the basic storage registers of both the Clipping Divider and the Channel Control. Each side of each card represents a 4-bit slice of data. Each of the two similar halves (A and B) contain up to 8 individual registers. In addition, the A side contains a counter, and the B side has gating allowing it to accept special input data.

7.2 Reference

E & S logic drawing No. 101107-600, Register Card Complete.

E & S logic drawing No. 101107-601, Register Card 2 x 5 Bit.

E & S logic drawing No. 101107-602, Register Card 2 x 8 Bit.

E & S block diagram No. 101101-900, PDP-10 Display Processor Registers and Busses.

7.3 Card Contents and Function

There are four possible input busses for the registers. One of the four is selected by four-way switches and clocked into the registers selected by a decoder. Register output is also enabled by an appropriate signal from a decoder. The output is inverted and passed to output switches which select either the true or compliment output. The selected input is also immediately available as output without having been registered.

The following description will follow the basic data path from the input to output.

7.3.1 Input

There are four basic input busses coming to the Register card:

1. The IN bus, which is unused in the processor.

2. The INA, INB bus carries true data from the right portion of the memory buffer register (MBR).

Either the IN or INA, INB bus is selected by a two-way switch (2-wide AND-OR-Invert gate). Enabling signals *EINB and *EINA determine the selection. Since the IN bus is unused, *EINA is tied to ground and *EINB to Vcc. The output of the two-way switches forms bus DA, DB which is available as immediate card output and also sent to the register input selection switches. The bus DA, DB is in compliment form.

3. The POA, POB bus contains data in compliment form from the open collector SB bus.

4. The NOA, NOB bus carries compliment data from the left half of the MBR (*MBR bus).

In addition, the output of the registers themselves can be recycled as input for the register input switches. This data is inverted by the output selection switches and, thus, sent to the input switches in compliment form.

7.3.2 Register Input Switches

These switches are made up of 4-wide AND-OR-Invert gates.

The switches select between (1) the DA, DB bus, (2) the POA, POB bus, (3) the NOA, NOB bus, or (4) the RA, RB output bus as input for the 4-bit registers. The Status Register and Instruction cards determine this selection. These signals are called *EPOA, *EPOB, etc., and are sent to both sides of the card. The selected input is inverted by the switches and becomes BUS1A, BUS1B, which is now in the true form and sent to the registers.

7.3.3 Register Selection and Function

The data of BUS1A, BUS1B are registered in only one of the registers on each side of the card. Selection is determined by the 4-bit binary number CODE0A - CODE3A, which is generated by the Instruction card. This number is decoded and the output ANDed with the clock pulse. The resultant signal clocks information into the selected registers. The registers are SM63's (or equivalents) which do not propagate their contents until enabled by an output enabling signal. The binary number CODE4A - CODE7A, also from the Instruction card, is decoded and enables the output of one of the registers. The register selection signals (both input and output) are sent to both sides of the card. For example, an input selection code of 0011 would clock data into register 3 (fourth from the top of the logic diagram) on both sides of the card. Similarly, an output selection code of 0010 would select the output of register 2 on both sides of the card. Either input or output selection codes equal to or larger than 1000 (i.e., the most significant bit is high) will

select no register. The outputs of all of the registers are tied together to form a common bus (BUS2A, BUS2B). Discrete pull up resistors and pull up resistors within the Register's IC's provide the necessary resistance so that register output will be in correct form. If no register output has been selected, BUS2A, BUS2B will contain all 1's.

The data on BUS2A, BUS2B is inverted and both the true and compliment information is presented to the output switches.

7.3.3 Output Switches

The output switches consist of 2-wide AND-OR-Invert gates. The ROM signals *ECOMP and *ETRU select either true or compliment data from BUS2A, BUS2B. The selection of true data means that if this data is recycled into the registers, it will be registered in true form and not that the output busses contain true information. *ETRU is tied to ground and *ECOMP is tied to Vcc in the processor.

7.3.4 Output

There are three basic output busses:

1. The OUTA, OUTB bus is taken directly from the OUT2A, OUT2B bus after inversions for buffering. OUTA and OUTB are sent onto the RMIB bus for use in push word and for loading the higher numbered registers of the processor. This bus contains all 1's if no input has been selected.

2. The DA, DB busses contain the compliment of data contained in either the IN bus or the INA, INB busses. They are

unused in the processor.

3. The data selected by the register output switches form the RA, RB busses. RA and RB are sent to the RA bus of the memory address register (MAR). The busses contain all 1's if neither true nor compliment outputs have been selected.

7.3.5 Clock

The Register card is designed to accept either true or inverted clock pulse. If a compliment clock is available, it is wired to C28, inverted, and jumpered on the back panel from C30 to C32.

7.4 Special Applications in the LDS-1 Channel Control

Several versions of the Register card exist, differing only in the number of components actually in place. The processor uses three 2 x 8 bit register cards. The A and B sides are used together so that each card makes up an 8-bit slice, with naturally only 2 bits used on the third card. Each card contains 8 registers numbered from 0-7 as shown on the block diagram of the registers in the processor (101101-900). The Register cards used do not have the counter circuitry components in place, as these are only used in the Clipping Divider.

INSTRUCTION CARD

8.1 Objective

The Instruction card deals directly with the instruction portion of each command interpreted by the processor. The card contains the 18-bit Instruction Register (IR) and logic intended to partially decode the instruction. The Instruction card contains the mode flip-flops of the processor and logic that generates the addresses for register selection.

The Instruction card also contains logic that helps in the microcode interpretation. The small offset numbers required for dispatching, as well as the DONE signal required for freezing the microcode in one step, are generated by logic on the Instruction card.

8.2 Reference

E & S logic drawing No. 101136-600, Instruction Card.

8.3 Card Contents and Function

The following discussion will first deal with the operation of the instruction register and the decoding. The mode flip-flops and their functions will then be examined. The logic generating the offset numbers for microcode dispatching, and finally, the register address selection logic will be discussed. This order follows the order of the sheets of the logic drawing.

8.3.1 Instruction Logic

The instruction register shown on sheets 1 and 2 of the logic drawing is made up of 18 D-type latches. Data on the MBR inputs to the Instruction card, which come from the left half of the Memory Buffer, are loaded into the register whenever the POKE IR signal is high. The true value of the instruction register contents are delivered to output pins (IR(0)-IR(17)) for use in push word and for transfer to the Repeat Status Register.

Bit 13 of the instruction register is connected to an exclusive OR circuit whose output (COND OK) is used during the conditional testing operations of group 2 instructions. If bit 13 contains a 1, the sense of the test to be made is inverted.

8.3.2 The Mode Logic

The Instruction card contains three J-K flip-flops to save the mode information of the processor. There are 8 possible combinations of mode flip-flop states:

FF CONTENTS	MODE	COMMENTS
000	PROG	The asynchronous inputs from CLEAR MODE and CLEAR EXFF are used in the "clear" step to put the processor into PROG mode.
001	PEEL	
010	REPT	Will revert to PROG mode when the repeat is finished.
011	REPT	Will revert to PEEL mode when the repeat is finished.
100	EXEQ	Will revert to PROG mode for next instruction.

101	EXEQ	Will revert to PEEL mode for next instruction.
110	REPT	When repeat is finished, it will do one instruction in EXEQ mode and then revert to PROG mode.
111	REPT	When repeat is finished, it will do one instruction in EXEQ mode and then revert to PEEL mode.

The repeat flip-flop may be asynchronously cleared by RCR(18) if CLEAR MODE is high. The execute flip-flop is cleared when CLEAR EXFF goes high. The mode flip-flops are loaded from the instruction register whenever POKE MODE goes high. The bits of the instruction register are decoded by the logic shown on the drawing just below the mode flip-flops as shown in the table below.

IR BITS 14-17	RESULTS
0000	Do nothing
0001	Go to PROG mode, clear EXFF and REPT
0010	Go to PEEL mode, clear EXFF and REPT
0011	Clear EXFF and REPT
01XX	Go to REPT mode, clear EXFF (XX indicates PROG and PEEL modes as above)
10XX	Go to EXEQ mode, clear REPT (XX indicates PROG and PEEL modes as above)
11XX	Go to EXEQ and REPT (XX indicates PROG and PEEL modes as above)

The outputs of the mode flip-flops are gated onto output lines labeled PI for use in the push word. This mode information is used only for the "marked" push words. The outputs are enabled by the *IR4 bit of the instruction, which is the relevant bit.

8.3.3 Small Number Logic

The small binary number circuit (sheet 3 of the logic drawings) produces the offset numbers to be used in the micro-code dispatching. The dispatch in question is indicated by the 3-bit SN code from the ROM. These bits are decoded, driving 1 of 5 lines high and thus selecting one of the inputs generated by the circuitry of the small number selection logic. Each of the circuits that generate the small numbers is a priority chain which selects numbers according to the state of the inputs in a specific order. For example, the circuit shown on the left of the drawing operates in the following priority:

```
IF GP3 and *SC=0 then produce the number "3" ELSE
IF REPT then produce the number "4" ELSE
IF EXFF then produce the number "0" ELSE
IF PEEL then produce the number "1" or if PROG produce the
  number "2"
```

Each step down the priority chain picks up another ELSE term to lock it out in case of a higher priority event. The offset number selected is inverted by the selection switch to form the *EXAO outputs which are sent to the Microcode Control card. Test points are provided to allow seeing the numbers in true form.

8.3.4 Address Selection Logic

The logic shown on sheet 4 of the drawing is used to select the register of the processor to be used. At the bottom of the drawing, a 3-bit register comprised of D-type latches is shown. This register is entitled WHO in the algorithm flow diagram. WHO is loaded from the GRA inputs when the POKE PARAMETER input

is high. The output of the latches can be seen on test points B, C, and D. A single bit latch (test point E) is used to remember whether the register whose address is in WHO is incrememted or decrememted when passing through the Memory Address Register (MAR).

The selection switches at the top of the drawing select one of the possible addresses according to the commands of the microcode. These commands enable the lines shown at the extreme right and left of the drawing. One of the 4 possible selections of register address is specified by the microcode in the following manner:

1. By specific number contained on the GRA inputs. This input is enabled by selecting none of the other inputs.

2. By selection of the IR bits 9-12 which contain the register address given in the instruction. The selection is made by driving UIFR (use instruction for read) or UIFW (use instruction for write) high.

3. By selection of the contents of the WHO register. The selection is made by high signals on UPFR (use parameter for read) or UPFW (use parameter for write).

4. By selection of register 5. The selection of register 5 (for write only) is made by a high signal on U5FW (use 5 for write).

If the parameter register is selected for read (UPFR high), the output MARUPC is driven from the remembered value in the single bit latch, otherwise the MARUP input from the microcode is copied into MARUPC.

8.3.5 The DONE Logic

The DONE logic shown on sheet 5 of the logic drawing selects

one of four conditions for the DONE output. These four conditions are selected by uniar coding on the four microcode input lines as follows:

SELECTED LINE	MEANING OF MNEMONIC	CONDITION
RBRFED	Read Buffer Register Free Enables Done	RBR free
PRED	Pause Request Enables Done	Pause request
SZED	Small count Zero Enables Done	*SC=0 or settled
SED	Settled Enables Done	Settled

The settled line collects together information about all of the units in the pipeline. Settled true indicates that all units are finished with their current jobs and are in their input waiting states.

COUNTER REGISTER

9.1 Objective

The Counter Register card contains the Read Count Register (RCR) and Write Count Register (WCR) of the Channel Control. The RCR is principally used to specify the number of data fetches to be done in repeat mode. The RCR is incremented each time the Read Address Register is incremented. When teh RCR becomes positive, the high-order bit drops to its zero state which resets the mode flip-flop on the Instruction card and causes the processor to drop out of repeat mode.

The WCR is normally used to count the number of times the Write address register is used for writing data into memory. By loading the WCR with some number, the size of the table written into memory can be limited. The high-order bit in its zero state is used to set the processor's stop flip-flop which, in turn, can cause computer interrupt.

9.2 Reference

E & S logic drawing No. 101133-600, Counter Registers.

E & S block diagram No. 101101-900, PDP-10 Display Processor.

9.3 Card Contents and Function

The counter registers are each made up of 18 J-K flip-flops. Data indicating the number of data fetches or the size of a

memory table may be loaded into the flip-flops via their set-clear pins. The flip-flops are concatinated in such a way that a count may be accomplished by enabling the clock pulse to the register. Data may also be read from the register by appropriate READ signals which allow the output of the flip-flops to be put on the open collector SB bus. When the register reaches a positive state, TEST BUSS and status register logic detect the zero state of the high-order bit. The following discussion will deal with the operations of the counter registers in the order specified above. Since the operations of the two counter registers are identical, the description is applicable to both registers.

9.3.1 Loading the Registers

Loading of the registers is accomplished by the signal R10LOAD for the RCR and R11LOAD for the WCR. These signals enable the input bus RMIB which is common to both of the registers. Each bit of the bus is inverted and the true and compliment signals are applied to the set-clear pins of the flip-flops, thus storing the data in true form.

9.3.2 Counting the Registers

The counting of the registers is accomplished by enabling the clock pulse which strobes the flip-flops. The J and K inputs of each flip-flop are tied together so that the flip-flop either remains in its previous state or is complimented. The low-order flip-flop (fed by RMIB (35) has its J-K inputs

tied to Vcc. Thus, wheever the flip-flop is storbed, it compliments. The next flip-flop in the chain is conditioned by the output of the low-order flip-flop. The remaining flip-flops are conditioned by the AND of the output and the J-K condition of the previous flip-flop. Thus, in order for the J-K condition of any flip-flop to be 1, the J-K condition for the previous flip-flop must have been one and the old value of the previous flip-flop must have been 0. In this way, the counters increment each time they are storbed until finally the positive state is reached.

The strobe signal for the registers is an AND of the clock pulse, COUNT RCR, and REPT MODE for the RCR and an AND of the clock pulse and COUNT WCR for the WCR.

9.3.3 Reading of the Registers

The data in the registers can be read by driving the *R10READ signal low for the RCR and by driving *R11READ low for the WCR. These signals enable the output of the flip-flops onto the open collector SB bus which is common to both registers.

9.3.4 Detecting the Positive State

When the counter has reached a positive state, the low-order bit drops to zero. The low-order bit for the read count register (RCR(18)) resets the mode flip-flop on the Instruction card and causes the processor to drop out of repeat mode. The low-order bit of the write count register (WCR(18)) sets the processor's stop flip-flop.

The high-order flip-flops for each register are also used to generate a "minus but not minus one" condition. This condition can be read onto the open collector TEST BUSS signal by driving TEST 10 high for the RCR and TEST 11 high for the WCR. In the J-CONDITIONS mode (JX high) of the group 2 instructions (CHANGE COND high), both registers are allowed to count once. There are, however, no such provisions for the K-CONDITIONS mode.

The signal *R14READ going low, which indicates the reading of the Status Register, allows the "minus but not minus one" condition to appear on the SB bus. Bit 26 indicates the state of the RCR and bit 27 the state of the WCR.

COMMAND AND DIRECTIVE

10.1 Objective

The Command and Directive card contains two registers which hold the directive to the processor and the low 16 bits of the directive that are to be passed to an external device.

10.2 Reference

E & S logic drawing No. 101135-600, Command & Directive Card.

10.3 Card Contents and Function

The directive register (DIR) is made up of 18 D-type latches. The directive to the processor is captured from the RMIB bus and loaded into the directive register at clock time, if the signal R12LOAD is high. The output of the directive register is read onto the open collector SB bus when the signal *R12READ goes low. The pull up resistors for this bus are also on the Command and Directive card.

When POKECOM is high, the high 16 bits of the directive register are loaded into the command register which is also composed of D-type latches. The compliment output of this register, labeled COM(18)-COM(33), is inverted by power drivers to provide sufficient drive and sent to the external devices being driven by the processor.

STATUS REGISTER

11.1 Objective

The Status Register card contains the Status Register (SR), associated test logic, and logic used to decode higher numbered register addresses (registers 10-17). The Status Register is used to store conditions that are tested in conditional instructions. The logic associated with the status register is used to select and enable the test conditions to be sent to the microcode control. The register address code for the higher numbered registers is decoded and used by logic on the Status Register card to select one of the registers for accessing.

11.2 Reference

E & S logic drawing No. 101141-600, Status Register.

11.3 Card Contents and Function

In the following discussion, we shall first deal with the Status Register logic, then the logic used to generate the TEST BUSS signal, and finally, with the register selection logic.

11.3.1 Status Register Logic

The 7 J-K flip-flops of the status register contain information about:

1. 4 program flags
2. HIT and AIC states from the Clipping Divider
3. Program STOP

These flip-flops are loaded through their asynchronous inputs by data on the RMIB bus (bits 18-21, 28, 29, 33) when a high signal on R14LOAD enables the clock pulse, in turn enabling the RMLB signals. The compliment value of the data on RMIB is applied to the clear pin. As low pulses are required to initiate both set and clear, data is stored in the flip-flops in true form. The inputs to the HIT and AIC flip-flops can also be controlled asynchronously by the Clipping Divider. Signals SETHIT, SETAIC, CLRAIC perform the functions indicated by their mnemonics at clipper clock time (i.e., when CLIPCL goes low). The HIT flip-flop is cleared when CLRHIT and CONOSETB go high. Because these two flip-flops are subject to such asynchronous set and clear operations an ambiguous result may be obtained if the status register is read just as the clipper changes one of these inputs. However, a synchronizer on the Microcode Control card resolves such conflicts when the flip-flops are being tested by the microcode.

The true outputs of all of the status register flip-flops are brought to output connectors to allow the signals to be sent to the microcode for testing. In addition, the contents of the status register may be read onto the open collector SB bus, which is common to all of the higher numbered registers, by driving the signal *R14READ low.

11.3.2 TEST BUSS Logic

The outputs of the flip-flops may also be selected as the test condition for the TEST BUSS signal. The test selection

signal, which determines which of the status register flip-flops to test, or alternately to test either the RCR or WCR "minus but not minus one" states, is generated by decoders cascaded to form a 1 of 17 (octal) decoder. This decoder works on instruction register (IR) bits 14-17 and develops the test selection signals TEST 0 - TEST 17 as follows:

TEST 0 - TEST 3	tests the program flags
TEST 10 - TEST 11	taken to output connectors to test RCR and WCR states
TEST 12 - TEST 13	tests the clipper conditions HIT and AIC
TEST 17	tests the program STOP flip-flop

The other tests are unused and available for use in system extensions.

11.3.3 Synchronization Logic

Connected with the program STOP flip-flop is logic intended to synchronize the start, stop, pause behavior of the processor. The four NAND gates in positions 74 and 33 form two simple latches that remember what sort of stop request has been made. The inputs to these flip-flops, CONOSETB, SET I/O STOP, CLRI/O STOP, STEP, and CLRPRIV, come from the PDP-10 I/O system. The signal STOP BUSS is an open collector bus that combines all conditions that can cause an interrupt. These signals include:

TUT TUT FORBID	(coming from clipper and indicating improper selection of slave scopes)
MEM TO MEM AND WCR 18	(limiting the size of table written into memory)
HIT AND STOP ON HIT	
PROG STOP	

The STOP BUSS signal is extended off of the Status Register card and other stop initiating signals are added. The J-K flip-flop in position 76.6 synchronizes the stopping of the processor with the processor clock.

13.3.4 Register Selection Logic

Shown on sheet 4 of the logic drawing is the register selection logic. This logic is intended to decode the register address for the higher numbered registers (i.e., those not on the Register card). The address codes are decoded 1 of 8 to produce the *R READ and R LOAD signals for each of the higher numbered registers. The R LOAD signals are inhibited if neither WR (write) nor WRIFOK (write if condition OK) and $*IR_{13}$ are high.

On the lower left of sheet 4 the gates are shown which must be enabled to combine the inputs of the lower registers with the inputs of the upper registers. It should be noted that *R15 READ will be enabled if either MA TO REG is enabled or if register address 15 is indicated by the address code. Thus, the memory address register may be accessed as register 15 of the processor.

REPEAT STATUS REGISTER

12.1 Objective

The Repeat Status Register (RSR) drives the high 13 bits of the command register. These bits are dynamically updated by the finite state machines of the RSR when the processor is in repeat mode, thus allowing several convenient drawing sequences to be executed.

12.2 Reference

E & S logic drawing No. 101142-600, Repeat Status Register Card.

12.3 Card Contents and Function

The repeat status register is register 13 of the processor and may be loaded immediately off of the RMIB bus and read onto the open collector SB bus and pushed onto a marked stack, as is the case with the other higher numbered registers. The repeat status register can also be loaded from the instruction register (IR) bits. The bits loaded into the RSR are decoded and sent to the command register. These bits may be changed by the finite state machines used for drawing and mode sequences. Counters used in multiple register transfers are also contained on the Repeat Status Register card.

12.3.1 Repeat Status Register Loading and Decoding

Data from the RMIB bus is loaded into the repeat status

register when the signal R13LOAD is high. Data from the instruction register is loaded into the RSR when ITORSR is high. The register is partially composed of D-type latches which are loaded at clock time, and partially of J-K flip-flops. The inputs to the J-K flip-flops are applied to the set-clear pins, but the input signals are gated with the clock pulse, so that these flip-flops are also loaded at clock time.

The output of the repeat status register flip-flops is decoded and drives the high 13 bits of the command register. The first 3 bits of the RSR (RSR 18-20) are used along with IR bits 5 and 6 (RSR 23-24) to generate the first two command bits. Command bit 0 is the load/fetch command to the Matrix Multiplier and bit 1 is the load/fetch command to the Clipping Divider. The next seven command bits are decoded from instruction bits 3-8 as shown below:

FUNCTION	COMMAND BIT					INSTRUCTION BITS 3,4 & 5
	0/1	2	3	4	5	(RSR 21, 22 & 23)
LINE	0	0	0	X	X	111, 010, 101
SETPOINT	0	0	1	X	X	110, 011, 100
BOX	0	1	0	X	X	000
DOT	0	1	1	X	X	001
RETRIEVE	1	X	X	0	0	10X AND GP3
LOAD	1	X	X	0	1	00X AND GP3
SINK	1	X	X	1	0	11X AND GP3
STORE	1	X	X	1	1	01X AND GP3

	COMMAND BITS 6,7 & 8			
SIZE	1	0	X	001, 000
RELATIVE	0	1	X	111, 011, 101, 001
				INSTRUCTION BITS 6,7 & 8
FROM	X	X	1	101
TO	X	X	0	000, 001, 010, 011, 100

12.3.2 Finite State Machines

The 3 RSR flip-flops which are loaded by instruction bits 3, 4, 5 -orm a finite state machine for generating the repeat mode drawing sequences. Similarly, instruction bits 6, 7, and 8 feed a finite state machine which generates the data mode sequences. These finite state machines change state when POKE FSM is high and system clock is present. In the repeat mode, the drawing instruction is repeated, but the finite state machines are poked which changes the drawing command in such a manner as to provide several convenient drawing sequences (The drawing instruction modes are explained fully in the E & S Users Manual, chapter 3).

12.3.3 Counters

Instruction bits 9 through 12 drive four RSR flip-flops which constitute a 4-bit up-down counter. The counter outputs are connected directly to command bits 9 through 12. These 4 bits make up the clipper register address for register data transfers (instruction group 3). In multiple register transfer mode, the counter counts down if instruction bits 3 and 4 are

one and zero (group three retrieve mode), otherwise it counts up. Counting is enabled by driving SHORT COUNT high. Instruction bits 14 through 17 drive a 4-bit down counter which is also enabled by SHORT COUNT and is used to count the number of data transfer made in multiple register transfer mode. The zero state of this counter is detected when SC=0 goes high, which terminates register data transferring.

13.3.4 Miscellaneous Commands

The last command register latch stores the privileged signal which allows the permit register of the Clipping Divider to be loaded. This latch is driven by the true output of a J-K flip-flop whose J input is the ROM signal J-PRIV and K input is the ROM signal K PRIV.

The SWAP bit needed by the clipper is stored in a flip-flop on the RSR card. The flip-flop is fed by ROM SWAP when POKE COM and CLOCK are high.

The RSR card also provides three drawing instruction outputs called DO DIRECT, DO INDIRECT, and DO INTERNAL. These outputs are decoded from the first three RSR bits. The three-bit binary codes 4, 5, and 6 generate these three outputs respectively.

64-WORD READ ONLY MEMORY

13.1 Objective

The 64-word Read Only Memory (ROM) cards are designed to provide a very fast read only memory which sends control and enabling signals to various parts of a system.

13.2 Reference

E & S logic drawing No. 101110-600, 64-Word Read Only Memory.

Bit specifications: Appendix, Section 20.4.

13.3 Card Contents and Function

The ROM contains 64 word lines and 144 bit lines which are gated together to form output signals. The basic layout on the printed circuit board will be discussed in conjunction with the selection paths and then component placement, and bit bundling will be explained as it deals with the specific content of the ROM cards.

13.3.1 Layout and Selection Paths

The printed circuit board is layed out with 64 word lines running vertically on the component side. Adjacent to these lines is a Vcc line to which unused circuit inputs are tied. A low signal from the ROM Driver card selects one of these word lines. If no word line is selected, the word lines con-

tain all 1's.

On the solder side of the board, 144 bit lines run perpendicular to the word lines. Where a certain word should select a specified bit line, a hole is drilled at the intersection of the two lines and plated through. These bit lines form "bundles" that make up the input of logic gates. The lines within a bundle are logically ORed together, which allows the selection of any specified output bit by several different words. The number of bit lines in any specified bundle is determined by the number of words that should select the associated output bit. To avoid sneak paths, only one hole may be drilled in any one of the 144 bit lines.

There is a column of integrated circuits on either side of the ROM. Horizontal bit lines, alternating in groups of four, feed a gate on one side and then a gate on the other side. This alternating pattern must be taken into account in the drilling of the holes.

On either side of the board there is a patchwork of horizontal and vertical lines which can be used as connection for an "extender" or for bringing the output of the gates on the upper portion of the card to output connectors. If one of the vertical lines is used as an "extender" it will have two holes, if used to bring output to the connector, the line will contain only one hole. "Extenders" are explained in the next section.

13.3.2 Component Placement and Bit Selection

Quad 2 input NAND or dual 4 input NAND integrated circuits

are used depending on the number of bit lines in the bundles involved. Dual 4 input AND gates serve as extenders if more than 4-bit lines are required for a bundle. The output of a 4 input AND provides one of the inputs for a 4 input NAND and thus allows up to seven bit lines to condition an output bit. The 2 input NAND gates may be used only in the lower two IC positions on each side of the board because only these positions have output pins 3 and 11 wired. If outputs 3 and 11 are to be used in these positions, appropriate holes must be drilled in the patchworks on the edge of the card to bring the output to a connector.

The original holes were drilled by an automatic milling machine running off a numerical data tape prepared by a PDP-9 program. Changes require drilling out the old hole and drilling and plating a new hole. The bit specifications and component placements are indicated on the build sheets of the different ROM boards. The boards are numbered historically starting with 01. The two-digit assignment code is printed as the last two digits of the dash number on top of the board. In addition, these numbers are drilled into the board in the following manner:

x t t t t t t t t t t x

x u u u u u u u u u u x

where x represents a hole always drilled and the tens digit is drilled in one of the positions marked t and the units are drilled in one of the positions marked u. The holes are read starting near the outside edge and counting towards the center of the board.

13.4 Use in the LDS-1 Channel Control

The word lines in the ROM correspond to the steps in the algorithm. Five ROM cards are used in the Clipping Divider. The word lines are wired in parallel, so that at any time, the same word line on all of the six cards is enabled.

ROM signals are sent to various parts of the processor. The signals are indicated as circles on the back panel block diagrams (101102-900).

The actual ROM bits are shown in Section 20.4 of the appendix. The drilling and component placement is shown on the PDP-9 printouts that make up Section 20.5 of the appendix.

ROM DRIVER

14.1 Objective

The ROM Driver receives address data from the program counter on the Microcode Control card and with this data selects and drives one of the word lines of the 64-Word ROM.

14.2 Reference

E & S logic drawing No. 101111-600, ROM Driver.

14.3 Card Contents and Function

The input data for the ROM Driver comes from an 8-bit ROM address register located on the Microcode card. The upper two bits of address select between different ROM driver cards and are connected to inputs labeled P(1) and P(2). If only one card is used in a system, as is the case in the Channel Control, P(1) and P(2) are tied to Vcc. The lower six bits of address are decoded onto twelve lines by the Microcode Control card. Each two bits of the address contained in the ROM address registers is represented by four lines. Thus, there are three groups of four address lines (labeled ROMAS), which make up the input data for the ROM Driver. All of the ROMAS lines are available at test points for ease in checking.

The logic involved in word selection is most easily understood if one considers the box of 64 NAND gates on the

logic drawing as a four by four matrix where each element in turn consists of four gates. The address signals ROMAS0-ROMAS3 select one of the four gates in each element of the matrix. The signals ROMAS00-ROMAS30 drive one of the four rows of the matrix and the signals ROMAS000-ROMAS300 select one of the columns of the matrix.

To determine which gate is selected for a given ROM address, one should consider the low six bits of the address to be a number base four, i.e., digit grouping of two bits. Then the signals ROMAS0-ROMAS3 select the low-order digit, ROMAS00-ROMAS30 SELECT the middle digit, and ROMAS000-ROMAS300 select the high-order digit. The intersection of these signals indicates the gate being selected. For example, to determine which gate is selected for address 41, base 10, one should convert 41 to a base four number. This results in the number 221_4 which indicates that ROMAS200, ROMAS20, and ROMAS1 will be selected. These three signals intersect at the gate whose output is bug 43, pin 8, and feeds connector 58.

In the case that more drive is needed for a given ROM word line, this can be obtained by wiring the corresponding ROMAS address lines to the inputs of one of the four extra NAND gates labeled HA, HB, HC, and HD.

MICROCODE CONTROL CARD

15.1 Objective

The Microcode Control card provides a program counter which supplies the ROM Driver with address information, thus controlling the 64-Word ROM cards. In addition, the card contains two flag flip-flops, two test networks, an adder, and a save register which allows one level of subroutining in the ROM.

15.2 Reference

E & S logic drawing No. 101112-600, Microcode Control.

15.3 Card Contents and Function

Two flip-flops and a group of AND and NOR gates test various conditions indicated by signals from the Directive and Done cards and enabled by ROM signals. The results of these tests form signals T and U and their compliments. These test signals and three 2-bit ROM bytes (YES, NO, SEL) are gated together to determine JUMP or COUNT instructions to the program counter. An adder adds ROM signals ADDR with (1) *EXAD signals from the Instruction card or (2) the contents of the Microcode Control's save register or (3) 0. The resulting eight bits from the input for the program counter (ROM address register). The output of the program counter is decoded and sent to the ROM Driver to select ROM word lines and stored in a save register to be recycled to the adders. Output is also

sent to the Clock card for use in the I/O synchronization.

The contents of the program counter are thus conditioned by all of the signals coming into the Microcode Control. Possible operations generated by this input data are: (1) do nothing, (2) add one to the value of the program counter, (3) move the 8-bit ROM address into the program counter, (4) add the ROM address and the eight *EXAD bits from the Instruction card to form new contents of the program counter, (5) add the ROM address and the contents of the save register to form new contents for the program counter.

The following discussion will describe the functions of the Microcode Control card in the order in which they were mentioned in the first paragraph of this section.

15.3.1 Test Logic

The test logic is used to generate T and U signals and their compliments. These signals are then used as conditioning signals for instructions to the program counter. The T signals are generated by a 4 x 4 matrix of 4 input AND gates. One input of each gate is tied to Vcc. Another is tied to an input connector signal which indicates some condition of the system. These inputs are labeled TT. ROM signals CI20, CI40, CI100, and CI200 select rows of the matrix, while the ROM signals CI1, CI2, CI14, and CI10 select columns. The intersection of a selected column and a selected row enables the gate that forms that element of the matrix. If care is taken to avoid logical conflicts, more than one condition can

be tested simultaneously. Tested signals are logically ORed together to form the signals T and *T.

The U signal is generated by tests of the flag flip-flops and four external conditions (UT). A positive result of any one test drives U high. The flag flip-flops consist of two J-K flip-flops. Clocking is controlled by an external input signal LOADFF. The J side inputs receive data from the Done card (JFL1, JFL2) and the K side inputs are fed by ROM signals (KFL1, KFL2). Both true and compliment outputs are used resulting in both high and low signals, each of which is tied to a 2 input AND gate. These gates are enabled by the ROM signals that select the columns of the T test matrix. The outputs are ORed together.

Four 2-input AND gates are also used to test the UT conditions. The UT signals constitute one input for each of the gates. The ROM signals which select rows of the T test matrix also enable these UT test signals. Again the results are ORed together.

15.3.2 Jump and Count Logic

A long chain of combinatorial logic determines the COUNT and JUMP commands. The YES, NO, and SEL bytes from the ROM, the T and U signals and their compliments, and a signal labeled D from the Instruction card, which indicates the done condition, are gated together in various combinations to generate the JUMP and COUNT signals.

These arrays show the conditions under which JUMP or COUNT

signals will be generated and which of the test NAND gates is activated.

15.3.3 Adder

Input to the adder is selected by a two-way switch (2-wide AND-OR-Invert gate) from either the *EXAD bits from the Instruction card or the contents of the save register. The other set of inputs is made up of the ROM signals ADDR. Selection is controlled by the YES bytes as follows:

YES(1)	YES(2)	RESULT
0	X	0 added to ADDR (All 1's are added and Carry In is initiated)
1	0	*EXAD added to ADDR
1	1	Save register contents (*R) added to ADDR

Three 4-bit adders and selection switches are used to make up an 8-bit "conditional sum adder". This arrangement, designed to increase speed, uses two adders to perform the addition of the upper four bits, one assuming a carry and one assuming no carry. The carry out of the lower bit adder selects the appropriate sum. The output of the adders (which is inverted for the upper four bits) forms the input for the program counter.

15.3.4 Program Counter

The program counter is made up of eight SN7471 (or equivalent) J-K flip-flops with AND-OR inputs. The inputs are arranged so that the flip-flops contain compliment information. A clear signal will reset each of the flip-flops so that the program counter contains 0. Clock pulses are sent to each

flip-flop. If COUNT is commanded, one is added to the contents of the program counter. If JUMP is commanded, the sum from the adders is registered in the program counter with appropriate adjustments for the upper four bits which are in compliment form. The output of each group of two flip-flops is sent in two bit bytes and their compliments (4 bits in all) to four NAND gates where it is decoded and inverted to form the ROMAS outputs for the ROM Driver and Clock cards.

15.3.5 Save Register

The compliment output of the flip-flops, which is the true value of the program counter since the flip-flops contain compliment information, is also sent to a save register, comprised of D-type latches, for recycling into the adders. The information is clocked into the latches when the ROM signal PUSH R is high, which enables strobing of the latches.

MEMORY TIMING AND CONTROL

16.1 Objective

The Memory Timing and Control (MTC) card serves as the interface between the processor and the PDP-10 memory interface. The processor treats the memory as a device with which it communicates. Communication is much like that between the processor and the Clipping Divider or Matrix Multiplier. When the processor requires a memory access, it merely raises a flag to the MTC card and sends a few bits saying why it wants to communicate with memory. It is then up to the MTC card to communicate with memory and control the memory buffer register (MBR) of the processor.

16.2 Reference

E & S logic diagram No. 101137-600, Memory Timing and Control Card.

16.3 Card Contents and Function

The MTC card operates in one of two modes as determined by a switch on the test point end of the card. The MAP mode is set when the switch is up; and the *MAP mode is set, of course, when the switch is in the down position.

In the following discussion, we will first examine the flip-flops that control the memory access requests and the signals generated to effect the requests. We will then turn

to the parity network, acknowledgement flags, I/O bus operations, and system master clear. After this, the write demon logic, and finally, the circuitry engaged if memory does not respond, will be discussed.

16.3.1 Access Request Logic

When the CYCLE FLAG (C34) goes from low to high, a racy pulse generator sets the MAP CYC flip-flop. This flip-flop allows the mapper to take a memory cycle before the processor gets its memory cycle. When the MAP CYC flip-flop is set, the GO signal is raised high if the MTC card is in MAP mode. This GO signal is sent to the mapper, which echoes back the OK signal. OK sets the PROC REQ flip-flop which, when set, clears the MAP CYC flip-flop. If the MTC card is not in MAP mode, when the pulse initiated by CYCLE FLAG sets the MAP CYC flip-flop, the MAP CYC flip-flop will set the PROC REQ flip-flop, which, in turn, clears the MAP CYC flip-flop.

If the ready flip-flop (made up of NAND gates in positions 64.12 and 63.3) is set and when either PROC REQ is set or MAP CYCLE REQ and MAP are high, *SIREQ will be driven low which sets the memory request flip-flop in the PDP-10 memory interface. When the request flip-flop in the memory interface is set, the signal IREQ is echoed back which clears the ready flip-flop and drives *SIREQ high again. When IREQ goes low again (which occurs when the PDP-10 has acknowledged the address), the signal *ADRACK goes low. A low signal on *ADRACK forces REQ FAST CLEAR to the mapper if in MAP mode, or clears the PROC

REQ flip-flop and sets the ready flip-flop if the MTC card is in *MAP mode.

The READ and WRITE signals to the memory interface are controlled by the PROC REQ flip-flop. If this flip-flop is clear, READ will always be high. Since the MAP CYC flip-flop clears the PROC REQ flip-flop in MAP mode, only read memory requests can be made by the mapper. If the PROC REQ flip-flop is set, either STACK WRITE (C25) or DATA WRITE (C23) will cause WRITE to go high. If neither of these signals are high, READ will be high. If READ is high, when IREQ goes from low to high, a 50nsec one shot (Monostable Multivibrator) will be triggered causing a pulse which clears the memory buffer register (MBR). Write request initiates a sequence which sets the MBR. If either STACK WRITE or DATA WRITE is high, when PROC REQ is set, a 200 nsec one shot is triggered which in turn triggers a 50nsec one shot. The 50 nsec pulse sets the MBR; the 200 nsec enables the *LOAD MBR or *LOAD PUSHW signals depending on the states of DATA WRITE and PUSH TO MBR respectively.

16.3.2 Parity Network

If a write has been requested, the NAND gate in position 12.3 (sheet 2 of the logic drawing) goes high and the 4 parity inputs from the memory buffer register are used to generate the MWBP parity bit. MWBP will be high for even parity. If a read is requested, the parity flip-flop is set by a pulse from the memory bus (MRBP). If the parity bit is 1, the NAND gate in position 12.3 will go low requiring an even number of ones on

the parity inputs from the MBR to keep the MWBP output low. If the parity bit is 0, the parity flip-flop will be cleared, 12.3 will be driven high, and an odd parity on the MBR bits will be required to keep the MWBP signal low. Parity can be ignored if IGN PAR is driven high which sets the ignore parity flip-flop. PERR can be driven high by a high signal on READ1 (a read request) and a high signal on MWBP if the ignore parity flip-flop is clear.

16.3.4 Acknowledge Flags

When a write cycle is complete, the PDP-10 sends a WRITM pulse (C96) which sets the ACK flip-flop and raises the ACK flag. When the read operation is complete, *RDRSG is pulsed low, which triggers a 125nsec one shot. On the trailing edge of this pulse R DONE goes from low to high and triggers another racy pulse generator (NAND gates in position 24). When this pulse is generated, one of three things may happen:

1. PARITY ALARM will go high and the parity alarm flip-flop will be set if parity alarm occurs.
2. *MAP CYC COMP will go low if the MTC card is in MAP mode.
3. ACK will go high and the acknowledge flip-flop will be set. When this occurs, CYCLE FLAG goes low.

16.3.5 I/O Bus Operations

DATAO CLRB from the PDP-10 generates a *SET MBR 2 pulse if the processor is PAUSED. Again, if the processor is PAUSED, DATAO SETB will load data from the I/O bus into the memory buffer register by driving the *LOAD IOB signal low which also

sets the IOMBR flip-flop. This flip-flop may be cleared by the ROM signal clear IOMBR at the next clock pulse.

16.3.6 Master Clear

The master clear one shot in position 50 is fired if a master clear signal is received from the PDP-10 interface MC 10 or if master clear is set by the switch on the control panel. Both true and compliment signals are used to clear conditions within the MTC card and sent to output connectors for clearing the rest of the processor.

16.3.7 The Write Demon

The write demon for up to 4 devices is located on the MTC card. Three communication signals for each device are provided. The device REQ signal sets the master latch when the latch is strobed. The clock pulse to the latch is enabled if either there is no request in process or WBR ACK goes high indicating that the request has been processed. On the next controlled clock pulse, the contents of the master latch is strobed into a slave latch. The outputs of the slave latches are tied to a priority chain which locks out a lower priority request if two requests are made at the same time. This priority chain consists of AND gates which are conditioned by the state of previous latches. When a request has been made, all lower priority requests are locked out and the WBR FULL flip-flop is set on the next clock pulse.

WBR FULL signals the processor that a write cycle has been

requested. The signal taken from the compliment side of the flip-flop is used to generate the *LOAD WBR signal which enables the loading of the data on the device input bus into the write buffer register (WBR) of the Memory Buffer Register card.

The request with priority generates a *DATA signal to the device making the request. This signal tells the device to put its output onto the common device bus. The *DATA signal enables two directive bits (WD(1X), WD(2X)) onto an open collector bus. These signals indicate the register within the processor that contains the address to be used for the memory write. The bits decode as follows:

WD(1X)	WD(2X)	REGISTER
1	1	DSP (Data Stack Pointer)
1	0	RAR (Read Address Register)
0	X	WAR (Write Address Register)

The acknowledge (ACK) signal is sent to the device whose request is being processed at the time of the WBR load.

16.3.8 NXM and MPX CLR Signals

When *SIREQ goes low, a 100 nsec one shot is triggered. On the trailing edge of the pulse IREQ should be low. If IREQ is high when the trailing edge of the pulse comes, the memory has not responded to the request. If the address is not acknowledged (ACKNR low) then an MPX CLR signal is sent to the multiplexer and the one shot is retriggered and the process repeated until IREQ drops low. If the address has been acknowledged, the signal NXM (no excitement memory) is driven high and the NXM flip-flop is set. This flip-flop can only be cleared by system master clear.

CLOCK CARD

17.1 Objective

The Clock card is used to generate and distribute clock pulses within a logic assembly. These clock pulses allow a machine to operate asynchronously from other devices with which it communicates.

17.2 Reference

E & S logic drawing No. 101117-600, Clock Card.

17.3 Card Contents and Function

The Clock card consists of the following logic functions:

1. Clock generation and distribution.
2. Clock PROCEED logic.
3. Control panel gating logic.
4. Input/Output synchronizing logic.

They will be discussed in that order.

17.3.1 Clock Generation and Distribution

The clock is generated by one of three time delay loop arrangements formed by a delay line and a group of Monostable Multivibrators (MM).

This DELAY and MM system has a natural period of oscillation dictated by the value of the delay (T_D) and the time con-

stants of the MM's in the loop.

A control signal named PROCEED (25.11) controls the repetition rate of the Clock in the following sense. If the PROCEED control signal remains high during a clock cycle, or if it comes high at the end of the cycle, the cycle will be repeated at the normal period of oscillations. If, however, PROCEED is low at the end of a clock cycle, the next clock cycle will not be initiated until PROCEED goes high again. The logic for PROCEED is explained in section 17.3.2.

Gate 54 controls opening of the DELAY LOOP, thus enabling or inhibiting the generation of the clock pulses. Figure 17.1 shows a timing diagram for the clock generator. Assume PROCEED has been low for a period of time longer than the natural Clock cycle. As soon as PROCEED goes high, gate 54.8 reinitiates a Clock cycle by going low which, in turn, makes one of the inputs to the delay lines go low (the delay line used is selected by C LONG, C NORMAL, C SHORT from the control panel). After a delay T_D through the delay line, plus delay through gates, 66.8 goes low. When 66.8 goes low, two things happen:

1. Input to gate 64.2 goes low. Gate 64 generates the clock pulse, which is thus controlled by 64.2 or by the delay through the selected delay line. The width of the pulse is thus the delay through the selected line plus an inherent 20 nsec nominal through gates.

2. The MM 65 is triggered, thus initiating a chain of MM's which control the clock rate.

The Mono-stable at 65 has three purposes:

1. To provide a delay between the fall edge of CLOCK and the occurence of LATE CLOCK.
2. To trigger LATE CLOCK at the falling edge of 65.
3. To trigger one of three MM's selected by the position of switch P SHORT, P NORMAL, P LONG in the Control Panel. These MM's select one of three possible clock cycles.

As long as MM 65 or MM 61, 62, or 63 is high, the wide AND gate and 54 is held low which inhibits the repetition of the clock cycle. The microprogram has the ability of increasing the natural period of the clock by bringing the LONG CYCLE signal at C72 high. The long cycle flip-flop is set at the fall of late clock. When long cycle is set, an additional MM (51, 52 or 53 as selected by the control panel switch) is triggered by the signal from the preceeding MM (61, 62, or 63). The additional MM delays repetition of the clock pulse for the entire duration of the pulse produced by the MM, thus elongating the natural frequency of the clock.

17.3.2 Proceed Logic

The proceed logic on the Clock card allows different assemblies within a system to operate asynchronously. The control wire named PROCEED, together with the input/output logic described in paragraph 17.3.4, permits an assembly to

stop the clock while waiting for the data transfers to other asynchronous units. Proceed is a function of the input/output waits and of the RUN signal from the control card. For any input channel, the PROCEED signal will be low as long as the ROM has requested an input wait and the transmitting device has not sent a DATA READY signal or the receiving device is still holding its acknowledge signal high. For any output channel, the PROCEED wire will be low as long as the ROM has requested output and the FLAG flip-flop is still high or the receiving device is still holding its acknowledge signal high. PROCEED will also be inhibited by a high signal on *RUN from the control panel.

17.3.3 Control Panel Logic Gating

A particular state of the machine and specified input/output waits can be selected by means of the switches on the control panel. When the device reaches this step or the selected I/O wait is reached, the *RUN line will be driven high or a pulse is produced at the BNC connector (SYNC) on the control panel.

The state of the machine is fed into the Clock card through the ROMAS lines from the Microcode Control card. This state is decoded in binary and sent onto output lines that are sent to the Lamp Driver which drives the state indicating lights on the control panel. The decoded state is compared with the state selected by control panel switches. The comparison is effected through two adders (in positions 16 and 17) which add

the data from the control switches and the ones compliment of the data sent from the program counter on the microcode control card (ROMAS). Coincidence is detected by a wide AND gate when the adder output lines contain all 1's.

The coincidence signal is clocked into the flip-flop in position 13 by the LATE CLOCK signal. If coincidence occurs, and if the STOP ON STATE switch on the control panel is on (C69), a STOPPED ON STATE signal is sent to the control panel (Lamp Driver) which in turn drives the *RUN signal high, thus impeding further clocking. The state of flip-flop 13 indicates coincidence of the state of the machine and the state indicated by the control panel switches. This signal is fed regardless of the state of the STOP ON STATE switch to the SYNC hub via the OR logic of gate 11.6.

The MISC TO SYNC signal is ANDed with the signal named DISCRETIONAL. When the MISC TO SYNC signal is high, the DISCRETIONAL signal is available for monitoring at the SYNC hub. If the STOPPED ON MISC switch from the control panel is on, a STOPPED ON MISC signal will be sent back to the Lamp Driver card if the AND conditions are satisfied by MISC TO SYNC and DISCRETIONAL.

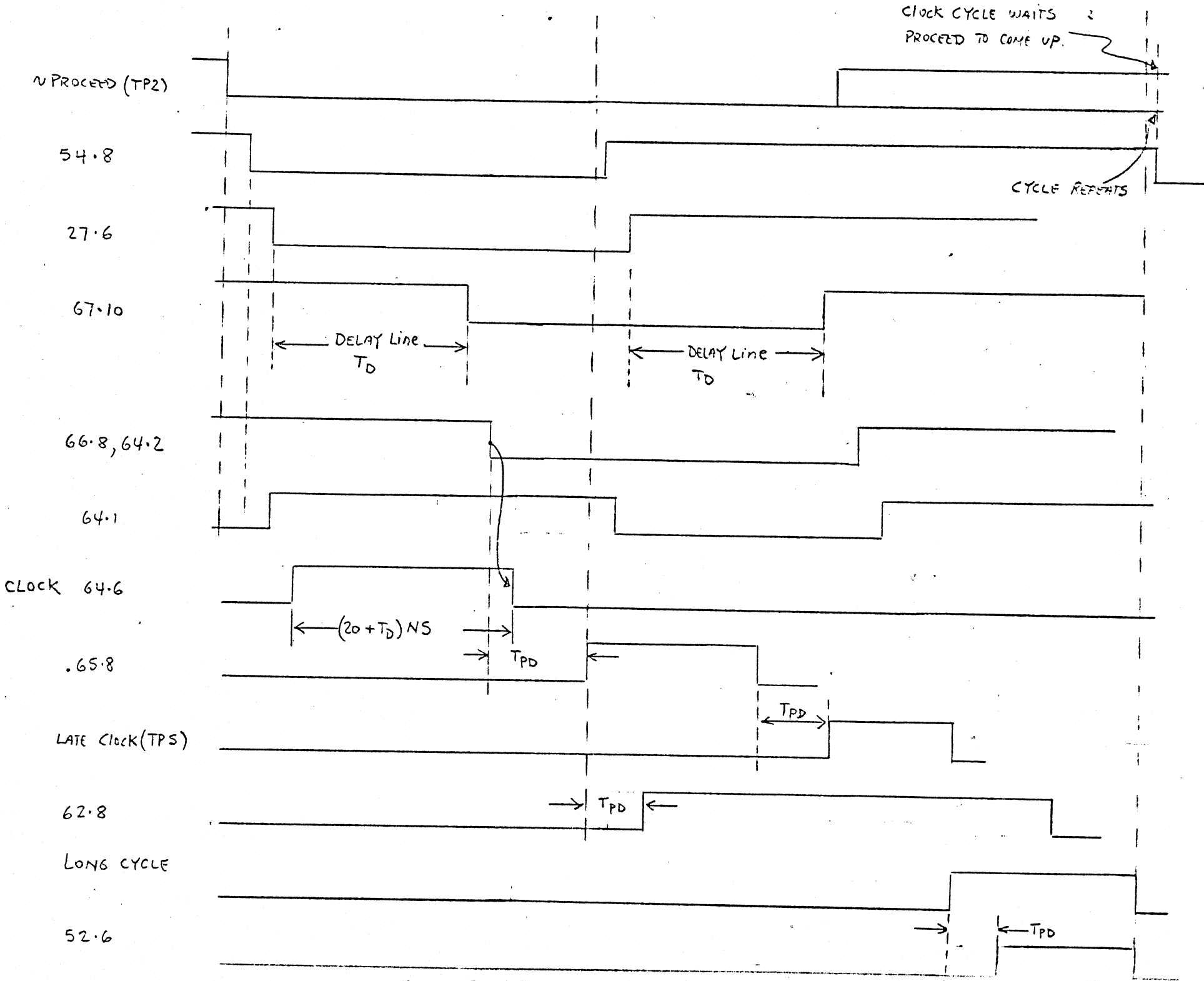
17.3.4 Input/Output Synchronization Logic

Figure 17.3.2 shows a timing diagram of the I/O sequences. The clock is equipped to handle 4 input and 4 output channels.

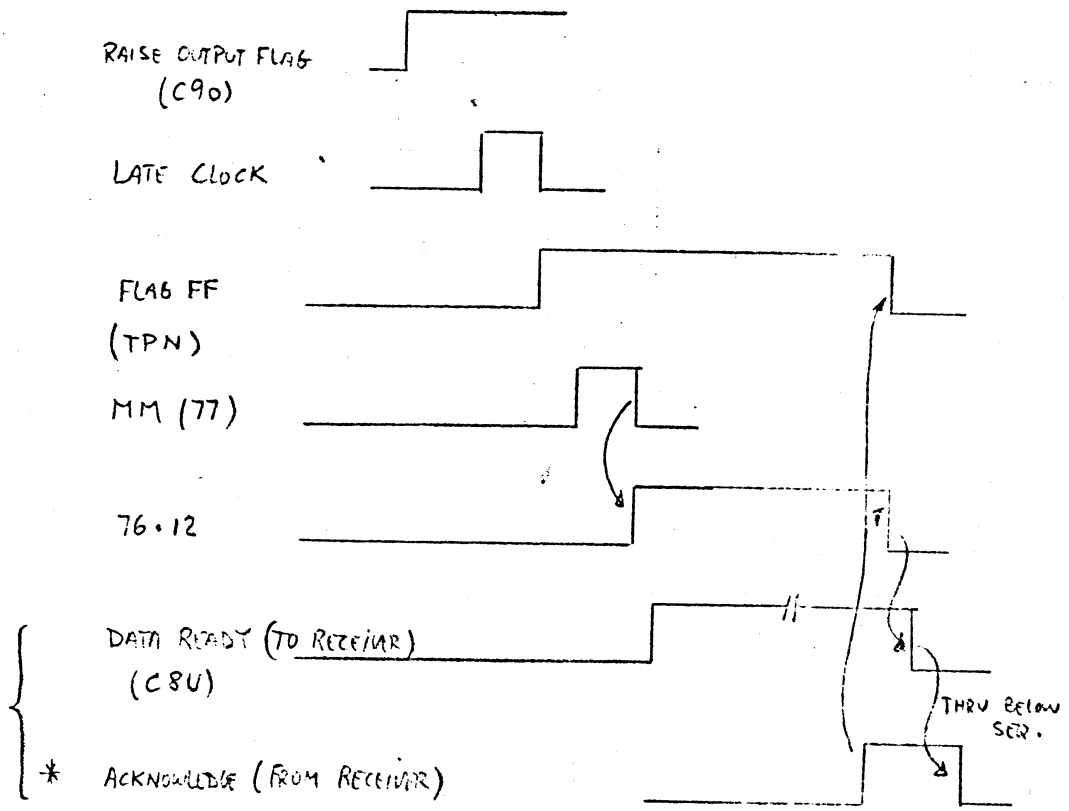
Each output channel consists basically of a J-K flip-flop, an MM for delay purposes, and another J-K flip-flop. The

first flip-flop can be considered the flag flip-flop. It is set by a RAISE OUTPUT FLAG signal. When set, it outputs a *FLAG signal which is fed to the clock inhibiting logic and drives PROCEED low. The output also triggers an MM which is used to ensure that data is stabilized on the lines before a DATA READY signal is sent to the device in question. The MM sets the next flip-flop and one input of a NAND gate whose other input is the output of the flip-flop. The signal is inverted once and then a DATA READY signal is sent. The ACK FROM signal that is answered back by the device accepting the input clears both flip-flops which allows clocking to proceed.

The input channel consists of a single J-K flip-flop. When a DATA READY signal is sent from the device transmitting the data, and when the ACK DATA TO signal from the ROM is raised high, an ACK TO signal is sent to the device in question. This signal clears the DATA READY signal from the outputing device, which in turn clears the flip-flop and drives ACK TO low.


CLOCK CYCLE WAITS
PROCEED TO COME UP.
~PROCEED (TP2)
54·8
CYCLE REPEATS
27·6
67·10
DELAY Line
T_D
DELAY Line
T_D
66·8, 64·2
64·1
CLOCK 64·6
$(20 + T_D)$ NS
T_{PD}
·65·8
T_{PD}
LATE CLOCK (TP5)
T_{PD}
62·8
LONG CYCLE
T_{PD}
52·6

OUTPUT SEQUENCE



INPUT SEQUENCE

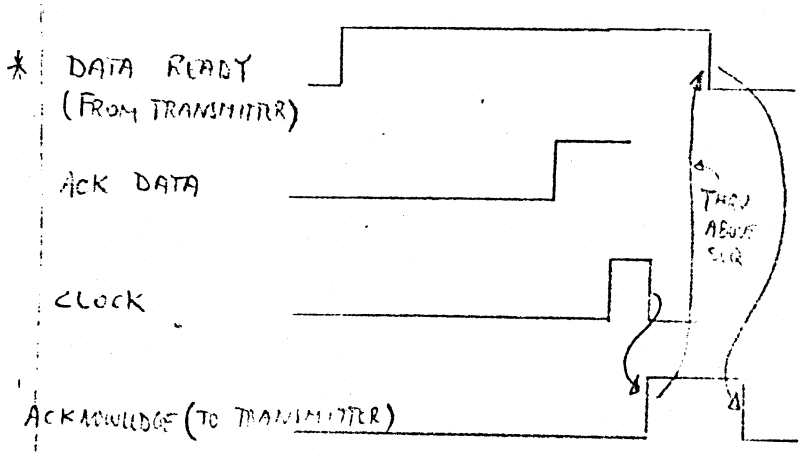


FIGURE 17.2

LAMP DRIVER AND CONTROL PANEL

17A.1 Objective

Two cards constitute the Lamp Driver Assembly. Their purpose is threefold:

1. to drive the control panel lights.
2. to provide interfacing between the control panel switches and other logic in the machine.
3. to allow stopping the clock under certain manual settings in the Control Panel.

17A.2 Reference

E & S logic diagram No. 101115-600, Lamp Driver.

NOTE: The logic diagram for the Lamp Driver includes both boards. IC designations of the form 1-xx refer to board -100; those of the form 2-xx refer to board -101. Connector designations are similar. 1Axx refers to the connectors at the top of board -100, 2Bxx refers to connectors at the bottom of board -101, etc.

E & S logic diagram No. 101117-600, Clock.

17A.3 Card Contents and Function

The Lamp Driver boards are located on the right side of the upper card cage. The board No. 101115-101 is for the most part the Lamp Driver proper. Board No. 101115-100 is the switch interface that interfaces between the switches of the control panel and thelogic of the machine; in particular, the clocking. The two boards will be discussed in the order mentioned.

17A.3.1 Driving the Control Panel Lamps

Board No. 101115-101 is mainly the Lamp Driver proper. All wiring of the Lamp Driver Board is between it and the control panel or the Clock card. All lights are driven by open collector drivers capable of sinking 80 ma at 4 volts. a 200 ohm bleeder resistor to ground allows current to pass through the lamps when the driver is off. This current tends to increase the life of the lamp and provides a slight glow when turned off which allows the determination of whether or not the lamp is still operational even if it is off.

The signals presented to the lights are as follows:

P3 to P8 (white)	- Indicates the state of the machine.
PROCEED (green)	- Indicates that there is no input-output transaction pending between assembly and other devices. If the clock is stopped, it must be because a manual setting of the switches caused it to stop.
STOPPED (red)	- Indicates that a manual setting of the switches has caused the clock to stop. The machine is stopped by one or more of the following conditions: 1. manual clock switch is on, 2. STOPPED ON I/O light is on, or 3. STOPPED ON STATE light is on.
STOPPED ON I/O (white)	- Indicates that the clock is stopped because one (or more) of the input-output wait switches is on, the STOP ON I/O switch is on, and the ROM is in I/O wait for the corresponding device. The switch that caused the STOP will have the corresponding light illuminated.
STOPPED ON STATE (white)	- Indicates that the clock is stopped because the STOP ON STATE switch is on and the ROM reached the state presented to the switches.
INPUT-OUTPUT WAIT (white)	- Indicates conditions as explained in STOPPED ON I/O.

17A.3.2 RUN Flip-Flop

Board No. 101115-100 is the Switch Interface and contains logic that controls the operation of the RUN flip-flop.

Sending of the STOP ON STATE, STOP ON I/O, or single step switches is synchronized by clock pulses to J-K flip-flops (1-41.6; 1-11.6; 1-41.1). This allows one to safely throw any of these switches on when the clock is running without producing faulty triggering of the clock.

17A.3.3 Single Step Clock Pulses

When the SINGLE STEP switch is on (1-41.14 low), the RUN flip-flop (1-11) clears and the *RUN wire at 1BI8 is high. As explained in the Clock card, the *RUN wire causes *PROCEED in the clock to go high, thus stopping the clock. As long as the SINGLE STEP switch is on, the only way to generate clock is by pressing either STEP LEVEL or STEP PULSE. STEP LEVEL (see clock card logic diagram No. 101117-600, sheet 1) at connector 95 by passes the normal clock generator and causes clock to go high as long as the STEP LEVEL switch is on. When the switch is thrown off, the *STEP LEVEL signal triggers MM 65 in order to initiate the late clock chain.

When STEP PULSE is activated, a monostable multivibrator (MM) (1-61) is triggered. During the pulse width of this MM, the RUN flip-flop is set causing *RUN (1BI8) to go low. The time constant in 1-61 was calculated so that the pulse width would not exceed even the shortest natural clock setting. A clock pulse is thus generated via the clock generator. When the

pulse in 1-61 ends, the RUN flip-flop goes immediately down again if the SINGLE STEP switch is not thrown OFF.

17A.3.4 STOP ON STATE

The STOP ON STATE switch permits stopping the machine as soon as the ROM reaches the address set in the State switches. When this occurs, the clock card brings *STOPPED ON STATE (2B11) low. This signal is fed to the RUN flip-flop reset (1-11.3), thus motivating *RUN (1BI8) to go high before the next clock cycle is initiated. In order to leave this stopped state, the STEP LEVEL or STEP PULSE switch has to be activated, producing clock as stated before.

17A.3.5 STOP ON I/O

The STOP ON I/O switch enables the INPUT WAIT, OUTPUT WAIT switches for any of several devices to stop the machine whenever a ROM step is reached that calls for an I/O WAIT for a device and the machine is still waiting for I/O from that device(s). Operation of the RUN flip-flop and procedure to leave this STOPPED condition are as described in section 17A.3.4.

17A.3.6 Rotaries

All other switches, except the STATE switches and the clock pulse rotaries are latched to assure proper on/off conditions of the logic signals.

Note that all switches (except the clock pulse rotaries)

are connected via a resistor to Vcc. All these switches have their common to ground.

The rotaries, however, have their common to Vcc. The contacts connect Vcc via a 100 ohm limiting resistor to the connector points in this card, where they are connected to ground via a 500 ohm resistor. The voltage at the input to the gate is a nominal 3.5 volts.

The purpose of the CLOCK PULSE WIDTH rotary is to enable one of three possible clock widths in the clock card. The center position is the normal operating position, while the other two decrease or increase the width with respect to the normal.

The PULSE INTERVAL rotary is also a three-position switch which allows increasing or decreasing the clock period with respect to the normal setting.

17A.3.7 SYNC

A BNC hub is provided in the control panel to monitor or SYNC the scope on one of several selected signals within the machine. The signals that drive the BNC hub are as follows:

1. If the CLOCK TO SYNC switch is on, the CLOCK is switched to the BNC connector.

2. If any of the INPUT/OUTPUT WAIR switches are on, the I/O WAIT signal for that device is switched to the BNC.

3. Any time a coincidence occurs between the position of the STATE switches and the state of the machine, the COINCIDENCE signal (see Clock Logic diagram No. 101117-600) is switched to the BNC.

4. When the MISC TO SYNC switch is on and there is

a signal wired (or jumped) to connector (C27) named DISCRETIONAL in the Clock card; this signal is switched to the BNC.

PDP-10 MEMORY INTERFACE

18.1 Objective

The memory interface provided between the processor and PDP-10 memory is a standard interface through a multiplexor as described in the PDP-10 interfacing manual.

18.2 Reference

E & S logic diagram No. 101138-600, PDP-10 Memory Interface.

18.3 Contents and Function

When the internal request signal goes to ground (*SIREQ from the processor), the internal request flip-flop is set and REQN is driven through a B683 to the multiplexor. The request signal is then sent back to the processor to indicate that the request has been made. When the request is granted, ACKNR goes to -3V generating the acknowledge signal. ACK puts the contents of the memory address register (MAR) onto the memory bus through B683. The READ and WRITE signals are also placed on the memory bus by the ACK signal. The ACK signal also gates all of the memory interface signals from the processor into the memory interface.

When ACK is received by the processor, the internal request flip-flop is cleared. If a write cycle was requested, the memory bus is pushed by WRITL and WRITR signals through W102. The

WRITM signal is generated to signal the processor that the write cycle has been completed. WRITL, WRITR, and WRITM are all generated by address acknowledged through B611's. All signals except the MA and MB signals that are sent to the processor are level converted to or from TTL logic levels by W512's or W607's located in the memory cage. These signals include:

WRITM	MWBP	READ
SIREQ	MRBP	WRITE
IREQ	RDRSG	IPAR
ACKNR		

PDP-10 I/O BUS INTERFACE

19.1 Objective

The E & S processor is interfaced to the PDP-10 in the manner suggested by the Interface Manual for the PDP-10.

19.2 Reference

E & S logic drawing No. 101139-600, PDP-10 I/O Bus Interface.

19.3 Contents and Function

Priority interrupt levels are placed into flip-flops from the I/O bus bits 33-35 on a CONO instruction. These bits are decoded 1 of 7 and placed on the interrupt lines when EPI goes low.

The IOS bits are decoded for the mapper and the display. The display is device 130 and the mapper 134. The decode is gated with DATAO SET, DATAO CLR, CONO SET, CONO CLR, DATAI and CONI signals to control data and conditional transfers. Conditions are gated onto the I/O bus when the CONI instruction is directed to the display. This is accomplished through B163's. W 107's serve to receive data from the I/O bus and distribute it to the processor and the conditional logic. The conditional bits are as follows:

CONO BITS

18 - System clear. This has the same effect as the console I/O reset switch and the clear switches on the Clipping Divider and Channel Control. The clipper and processor are cleared and the processor is sent to the STOP state while the clipper is sent to the INPUT WAIT state. Two successive clears are necessary if the clipper has been operating in the 3D mode. The clipper will not finish the line it is working on, nor will the processor complete its instruction. No information is lost in the processor.

19 - Allow Memory Alarm Interrupt. This allows non-existent memory and parity errors to cause the host computer to interrupt on the selected channel.

20 - Disallow Memory Alarm Interrupt.

21 - Unused.

22 - Allow Map/Protect Interrupt. This bit is used in connection with memory protection.

23 - Disallow Map/Protect Interrupt.

24 - Set I/O Stop.

25 - Allow Stop Interrupt.

26 - Disallow Stop Interrupt.

27 - Clear I/O Stop.

28 - Clear Program Stop.

29 - Clear Hit.

30 - Step.

31 - Unused.

32 - Allow Priority Interrupt Assignment.

33, 34, 35 - Priority Interrupt Assignment.

CONI BITS

0 - Program Flag 0

1 - Program Flag 1

2 - Program Flag 2

3 - Program Flag 3

4-17 - Unused

18 - Parity Alarm

19 - NXM Alarm (non-existent memory)

20 - Alarm Interrupt On

21 - Map/Protect Violation

22 - Map/Protect Interrupt On

23 - Unused

24 - Stopped and Ready

25 - Stop Interrupt On

26 - Memory To Memory Stop

27 - I/O Stop

28 - Program Stop

29 - Hit Stop

30 - Scope Select Violation Stop

31 - Unused

32 - LDS-1 Caused Interrupt (i.e. Interrupt has actually occurred)

33, 34, 35 - Priority Interrupt Assignment

APPENDIX

The appendix includes documents that define or describe the operation of the Channel Control. The following documents are contained:

20.1 Algorithm	The definition of the algorithm in in terms of PDP-9 macros. The ROM bits activated by each step are listed in octal codes that relate to the ROM bit descriptions that preceed the algorithm.
20.2 ROM bit Wiring	A listing of the ROM bits and their wiring.
20.3 Registers of the Processor	Shows the registers and busses of the processor
20.4 ROM 64 Word bit pattern	Shows the content of the ROM cards in terms of bit lines and word lines.
20.5 ROM 64 Word component placement	Defines the component placement in the ROM cards.

```
        .TITLE PROCAL
/BITS IN THE PROCESSOR ROM

/REGISTER CONTROL (REC1-RE11)
        /WRITE
        /WRITE UNLESS @ INHIBIT
        /POKE PARAMETER
        /USE PARAMETER REGISTER FOR READ
        /USE INSTRUCTION ADDRESS FOR READ
        /USE ADDRESS OF P2 FOR WRITE (5)
        /USE PARAMETER REGISTER FOR WRITE
        /USE INSTRUCTION ADDRESS FOR WRITE
        /3 BIT REGISTER ADDRESS
/PATH CONTROL (PAC1-PAC9)
        /SPARE
        /MBR LEFT INTO REGISTER
        /MBR RIGHT INTO REGISTER
        /HI OR LC REGISTER TO OUTPUT
        /SPARE
        /MA TO BUSS
        /SPARE
        /SPARE
        /MA+1 (IS PARAMETER)
/POKE CONTROL (PKC1 - PK12)
        /POKE FINITE STATE MACHINE
        /SHORT COUNT
        /POKE MODE
        /POKE MBR
        /POKE COMMAND REGISTER
        /POKE RSR (IR TO RS?)
        /POKE RBR
        /COUNT RCR
        /COUNT WCR
        /POKE CONDITIONS
        /POKE INSTRUCTION REGISTER
        /POKE MA
/FLAG CONTROL (FLC1 - FL11)
        /CLEAR MODE
        /RAISE OUTFLAG
        /CLEAR OUTFLAG
        /CLEAR EXEQ FLAG
        /SET F1
        /CLEAR F1
                /SET F2
        /CLEAR F2
        /CLR IOMBR
        /J PRIV
        /K PRIV
/IO CONTROL (IOC1 - IO11)
        /SPARE
        /STOPPED AND READY
        /ACKNOWLEDGE WRITE BUFFER REQUEST FLAG
        /REQUEST MEMORY CYCLE
        /WAIT ON MEMORY CYCLE
        /SWAP
        /POKE MAP BITS
        /BIT 1 (STACK WRITE CYCLE)
        /BIT 2 (DATA WRITE CYCLE)
        /BIT 3 (PROGRAM OR STACK READ)
>
```

```
        /BIT 4 (DATA READ OR EXECUTE)
/TEST CONDITIONS
        /200    SPARE   SPARE   SPARE   DO TWICE
        /100    STOP    WRITE REQ       PAUSE REQ       IOMBR
        /40     DO IND  DO DIR  DO INT  PUSH
        /20     @       3D      SHCNT=0 TEST COND
        /       10      4       2       1
/TESTING CONTROL (TSC0 - TS14)
        /(*SC=0 OR SETTLED) = DONE
        /SETTLED=DONE
        /STOP=DONE
        /*RBRF=DONE
        /3 DISPATCH SELECTION BITS
        /8 CONDITION BITS
/JUMP CONTROL
        /PUSH
        /2 YES BITS
        /2 NO BITS
        /2 SEL BITS
/JUMP ADDRESS
        /8 ADDRESS BITS

        .DEFIN ADDR,A
        A-BROMT-2/12
        .ENDM
        .DEFIN JUMP,A
        0
        21
        ADDR A
        .ENDM
        .DEFIN JUMPIF,A,B
        A
        20
        ADDR B
        .ENDM
        .DEFIN JUMPUN,A,B
        A
        21
        ADDR B
        .ENDM
        .DEFIN DISP,A,B
        A*400
        41
        ADDR B
        .ENDM
        .DEFIN CALL,A
        0
        121
        ADDR A
        .ENDM
        .DEFIN NEXT
        3
        0
        0
        .ENDM
        .DEFIN HOLDAC,C,W,S     /HOLD AND CALL
        C*4000+W
        130
        ADDR S
        .ENDM
>
```

```
.DEFIN HOLDAJ,C,W,A     /HOLD AND JUMP
C*4000+W
30
ADDR A
.ENDM
.DEFIN HOLDUJ,C,W,A     /HOLD UNLESS DONE, JUMP
C*4000+W
34
ADDR A
.ENDM
.DEFIN EXITIF,C,N
C
62
N
.ENDM
.DEFIN EXIT,N
0
61
N
.ENDM
.DEFIN JUMPOF,C,P
C
22
ADDR P
.ENDM
.DEFIN DISPIF,I,C,A
I*400+C
40
ADDR A
.ENDM
.DEFIN DISPUN,I,C,A
I*400+C
41
ADDR A
.ENDM
.DEFIN DO,R,P,K,F,M
R
P
K
F
M
.ENDM
.DEFIN NILL
0
0
0
0
0
.ENDM
.DEFIN POKE,A
0
0
A
0
0
.ENDM
.DEFIN FF,A
0
0
0
```

>

```
        A
        Ø
        .ENDM
        .DEFIN IO,A
        Ø
        Ø
        Ø
        Ø
        A
        .ENDM
        .DEFIN PATH,A,B,C
        A
        B
        C
        Ø
        Ø
        .ENDM
        .GLOBL BROMT
BROMT   BROMTL-BROMT-2/12
        12

CLEA    .ASCII .CLEA.                    0
        DO Ø,Ø,Ø,2655,400
        JUMP STOP

IOFF    .ASCII .IOFF.
        DO Ø,Ø,2,6,Ø     /IR AND CLEAR IOMBR, SET PRIV
        JUMP INDI                        1

/ORIGIN OF DISPATCH Ø

RAFE    .ASCII .RAFE.                    2
        DO Ø,1,1,200,Ø
        JUMP RAFF

PCFE    .ASCII .PCFE.                    3
        PATH 2,1,1
        JUMP PCFF

SPFE    .ASCII .SPFE.                    4
        PATH 3,1,1
        JUMP SPFF

GP3R    .ASCII .GP3R.                    5
        NILL
        DISP 2,SIST

DIDI    .ASCII .DIDI.                    6
        NILL
        JUMPUN 50,DIDV

/END OF DISPATCH Ø

DIND    .ASCII .DIND.                    7
        PATH Ø,1,1
        JUMP DINE

STOP    .ASCII .STOP.
        NILL                             10
        NEXT
```

```
PREP    .ASCII .PREP.                    11
        NILL
        HOLDAC 4,104,WRIT

PAUS    .ASCII .PAUS.                    12
        IO 1000
        HOLDUJ 2,101,IOFF

/ORIGIN FOR DISPATCH 3

MODE    .ASCII .MODE.                    13
        FF 1    /CLEAR PRIV
        DISPUN 0,110,RAFE       /UNLESS STOP

MODF    .ASCII .MODF.                    14
        NILL
        JUMP STOP

GRPI    .ASCII .GRPI.                    15
        NILL
        JUMP MODE

DOIT    .ASCII .DOIT.                    16
        POKE 100
        JUMP DIDI

RAFF    .ASCII .RAFF.                    17
        DO 2000,11,2,0,321
        JUMP INDI

PCFF    .ASCII .PCFF.                    20
        DO 2002,11,2,0,322
        JUMP INDI

SPFF    .ASCII .SPFF.                    21
        DO 2003,11,2,0,322
        NEXT

INDI    .ASCII .INDI.                    22
        PATH 2005,100,0
        DISP 1,OPCI

/ORIGIN FOR DISPATCH 1

OPCI    .ASCII .OPCI.                    23
        PATH 2105,40,0
        JUMP OPCA

LOST    .ASCII .LOST.                    24
        PATH 1400,101,100
        DISP 2,SIST

SIRT    .ASCII .SIRT.                    25
        PATH 1406,100,100
        DISP 2,SIST

COND    .ASCII .COND.                    26
        NILL
        HOLDAC 4,104,WRIT
>
```

```
/END OF DISPATCH 1

CONA    .ASCII .CONA.
        POKE 4
        JUMPUN 21,MODE

CONB    .ASCII .CONB.
        PATH 2015,40,0
        JUMP MODE

OPCA    .ASCII .OPCA.
        PATH 1010,100,0
        JUMPUN 41,CHMD

OPCB    .ASCII .OPCB.
        PATH 3,0,1
        NEXT

OPCC    .ASCII .OPCC.
        PATH 2003,10,0
        NEXT

OPCD    .ASCII .OPCD.
        DO 5,40,400,0,330
        NEXT

CHMD    .ASCII .CHMD.
        POKE 1000       /POKE MODE
        DISP 3,MODE

/ORIGIN FOR DISPATCH 2

SIST    .ASCII .SIST.
        NILL
        HOLDAC 1,104,WRIT

SISD    .ASCII .SISD.
        DO 0,0,2200,1000,0
        JUMP SISA

LORT    .ASCII .LORT.
        NILL
        CALL DATA

/END OF DISPATCH 2

LORU    .ASCII .LORU.
        POKE 2000
        JUMP MODE

DINE    .ASCII .DINE.
        DO 2000,11,24,0,321
        NEXT

DINF    .ASCII .DINF.
        PATH 2404,201,0
        NEXT

DING    .ASCII .DING.
>
```

27
30
31
32
33
34
35
36.
37
40
41
42
43
44

```
        PATH 2005,100,0
        CALL DATA

DINH    .ASCII .DINH.                                   45
        PATH 405,1,4000 /P2 => WHO
        CALL DATA

DINI    .ASCII .DINI.                                   46
        POKE 4000       /POKE FSM
        JUMP MODE

DIDV    .ASCII .DIDV.                                   47
        POKE 20
        JUMPIF 42,DIDU

DDIR    .ASCII .DDIR.                                   50
        PATH 400,1,0
        CALL DATA

DDIT    .ASCII .DDIT.                                   51
        POKE 4000
        JUMPUN 201,MODE

DIDU    .ASCII .DIDU.                                   52
        NILL
        HOLDAC 1,104,WRIT

DINT    .ASCII .DINT.                                   53
        DO 0,0,200,1000,40
        JUMP MODE

/DATA WRITING SUBROUTINE.  CALL WHEN WRITE REQUEST IS ON.
/CLEAR F1 AND F2 BEFORE ENTERING.

WRIT    .ASCII .WRIT.                                   54
        NILL
        DISP 4,DSPW

/ORIGIN OF DISPATCH 4

DSPW    .ASCII .DSPW.                                   55
        PATH 6,1,1
        NEXT

DSPA    .ASCII .DSPA.                                   56
        DO 2006,11,3,0,324
        JUMP WRIA

RARW    .ASCII .RARW.                                   57
        PATH 0,1,1
        NEXT

RARA    .ASCII .RARA.                                   60
        DO 2000,11,0,0,324
        JUMP WRIA

WARW    .ASCII .WARW.                                   61
        DO 1,1,11,0,0
        NEXT
```

>

```
/END OF DISPATCH 4

WARA    .ASCII .WARA.                          62
        DO 2001,11,0,0,324
        NEXT

WRIA    .ASCII .WRIA.
        IO 400                                 63
        EXITIF 4,0

WRIB    .ASCII .WRIB.
        NILL                                   64
        JUMPOF 2,DATD

/DATA FETCHING SUBROUTINE.  SET WHO AS YOU ENTER.

DATA    .ASCII .DATA.                          65
        FF 112
        HOLDAJ 1,104,WRIT

DATB    .ASCII .DATB.                          66
        DO 233,0,201,0,0
        JUMPUN 24,DATF

DATC    .ASCII .DATC.                          67
        DO 2020,10,40,1000,321
        NEXT

DATD    .ASCII .DATD.                          70
        FF 123
        HOLDAJ 1,104,WRIT

DATE    .ASCII .DATE.                          71
        DO 200,0,1,0,0
        NEXT

DATF    .ASCII .DATF.                          72
        DO 2020,10,40,1050,321
        EXIT 1

SISA    .ASCII .SISA.                          73
        NILL
        HOLDAC 10,104,WRIT

SISB    .ASCII .SISB.                          74
        NILL
        JUMP MODE

BROMTL 0

        .END
END OF FILE REACHED BY:
P 60
>
```

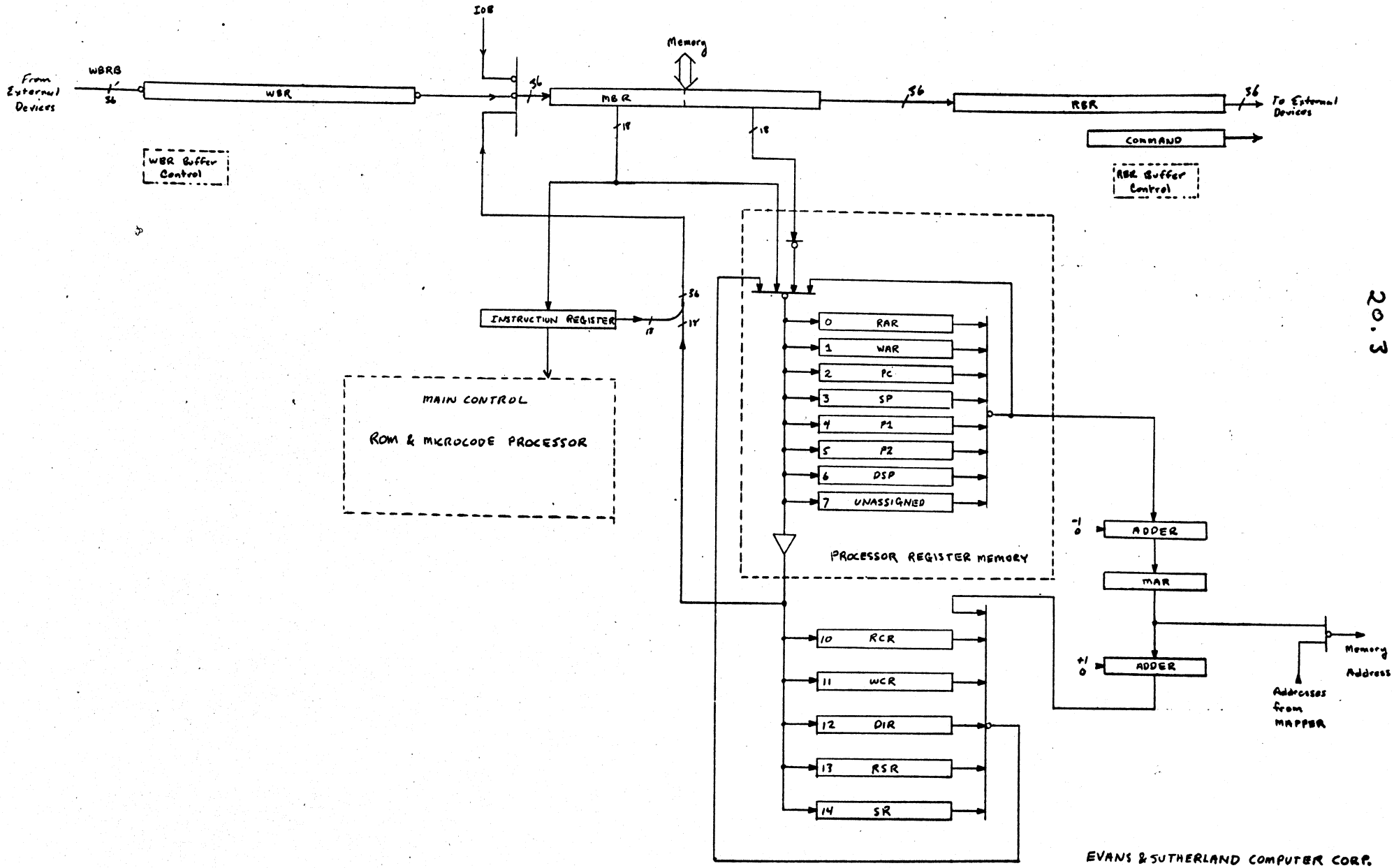
PROCESSOR ROM WIRE LIST

ROM NAME	WIRE NAME	ROM CONNECTION	CARD CONNECTION
REC1	WR	4.14	33.43
REC2	WRIFOK	1.11	33.41
REC3	POKEPARAMETER	5.11	11.19
REC4	UPFR	5.90	11.14
REC5	UIFR	5.88	11.15
REC6	U5FW	NOT USED IN THE ROM	
REC7	UPFW	5.89	11.21
REC8	UIFW	5.82	11.22
REC9	GRA(1)	5.86	11.16
RE10	GRA(2)	5.12	11.17
RE11	GRA(3)	2.11	11.18
PAC1		SPARE ROM BIT	
PAC2	*ENO	5.92	11.4
PAC3	*EDA	6.84	11.7
PAC4	REGTOREG	1.88	33.47
PAC5		SPARE ROM BIT	
PAC6	MATOREG	4.11	33.79
PAC7		SPARE ROM BIT	
PAC8		SPARE ROM BIT	
PAC9	MARUP	3.84	11.80
PKC1	POKEFSM	1.82	31.96
PKC2	SHORTCOUNT	5.87	31.95
PKC3	CHANGEMODE	5.91	11.53
PKC4	PUSHMBR	5.81	13.39
PKC5	POKECOM	1.87	29.42
PKC6	ITORSR	1.81	31.82
PKC7	LOADRBR	5.8	42.46
PKC8	COUNTRCR	1.15	35.15
PKC9	COUNTWCR	5.15	35.79
PK10	CHANGECOND	5.7	33.68
PK11	POKEIR	1.9	11.58
PK12	POKEMA	4.86	37.10
FLC1	CLEARMODE	5.9	11.35
FLC2	RAISEDEVFLAG	2.9	10.92
FLC3	CLEAR OUTFLAG	5.6	NOT USED IN PROCESSOR
FLC4	CLEAREXFF	5.16	11.54
FLC5	JF1	5.5	8.51
FLC6	KF1	5.10	8.50
FLC7	JF2	3.90	8.47
FLC8	KF2	2.88	8.46
FLC9	CLRIOMBR	3.87	13.18
FL10	JPRIV	3.16	31.27
FL11	KPRIV	3.81	31.29
IOC1	STOPPED	4.90	NOT USED IN PROCESSOR
IOC2	PAUSED	4.87	13.20
IOC3	WBRACK	4.5	13.48
IOC4	RAISEMEMFLG	4.83	10.90
IOC5	OUTPUTWAITA	5.13	10.65
IOC6	SWAP	4.9	31.28
IOC7	POKEMAPBITS	4.85	NOT USED IN PROCESSOR
IOC8	STACKWRITE	4.89	13.25
IOC9	DATAWRITE	2.82	13.23

>

IO10	PROGRAMOR STACK READ	4.81	NOT USED IN PROCESSOR
IO11	DATAREAD OR EXECUTE	2.16	NOT USED IN PROCESSOR
TSC0	SZED	4.82	11.40
TSC1	SED	4.8	11.39
TSC2	PRED	4.16	11.38
TSC3	RBRFED	2.10	11.37
TSC4	SNA	4.7	11.24
TSC5	SNB	3.82	11.25
TSC6	SNC	4.10	11.26
TSC7	CI200	3.10	8.56
TSC8	CI100	5.83	8.29
TSC9	CI40	3.15	8.16
TS10	CI20	4.92	8.43
TS11	CI10	4.88	8.52
TS12	CI4	4.84	8.49
TS13	CI2	4.12	8.48
TS14	CI1	3.9	8.45
JUC1	PUSHR	5.85	8.7
JUC2	YES(1)	5.84	8.6
JUC3	YES(2)	1.13	8.5
JUC4	NO(1)	5.14	8.38
JUC5	NO(2)	4.6	8.37
JUC6	SEL(1)	4.15	8.39
JUC7	SEL(2)	1.86	8.40
JUA1	NOT USED IN ROM OR PROCESSOR		
JUA2	NOT USED IN ROM OR PROCESSOR		
JUA3	ADDR(3)	6.14	8.24
JUA4	ADDR(4)	6.85	8.23
JUA5	ADDR(5)	2.14	8.82
JUA6	ADDR(6)	3.85	8.81
JUA7	ADDR(7)	2.85	8.80
JUA8	ADDR(8)	3.14	8.65

END OF FILE REACHED BY:
P 60
>



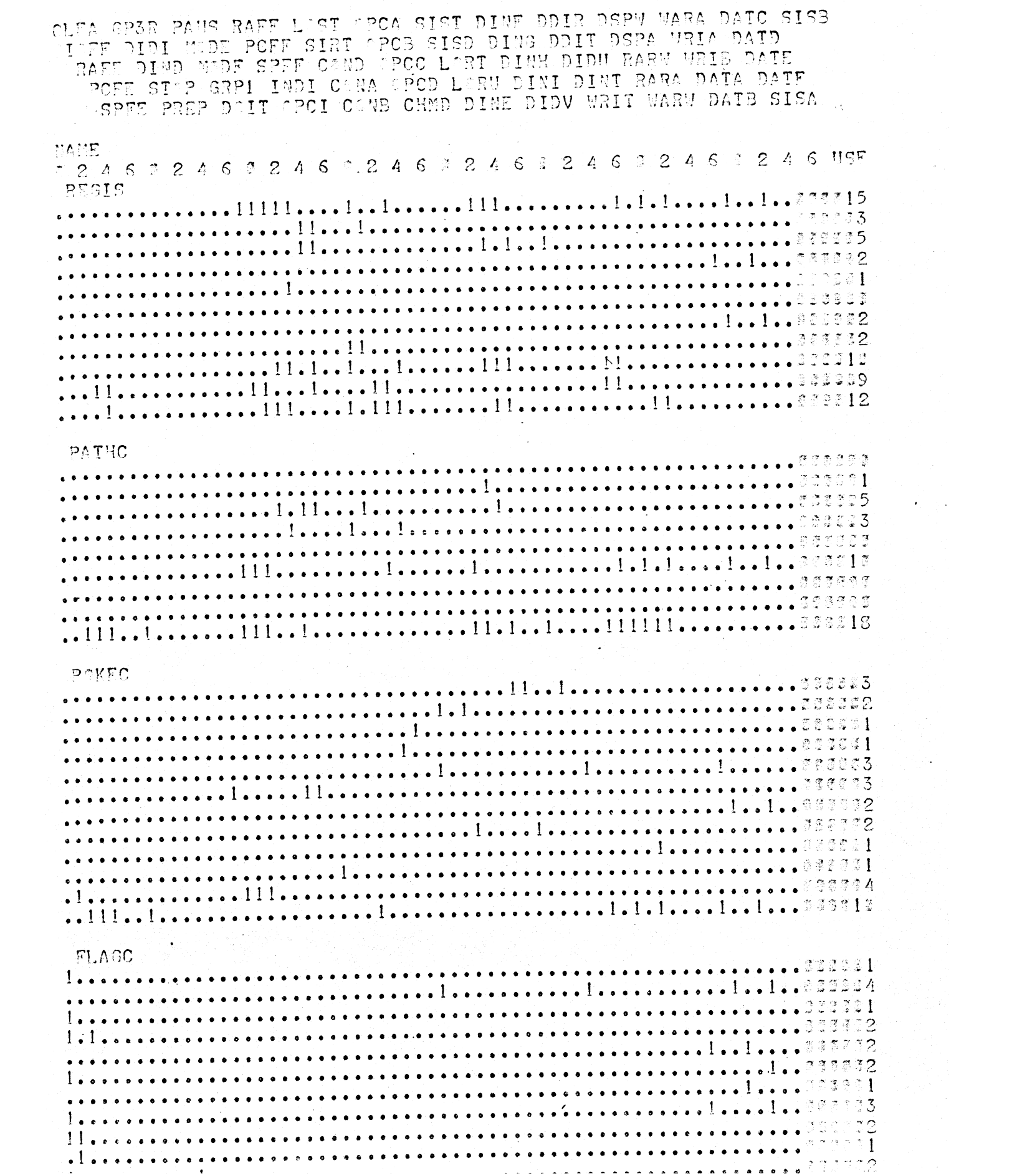
20.3

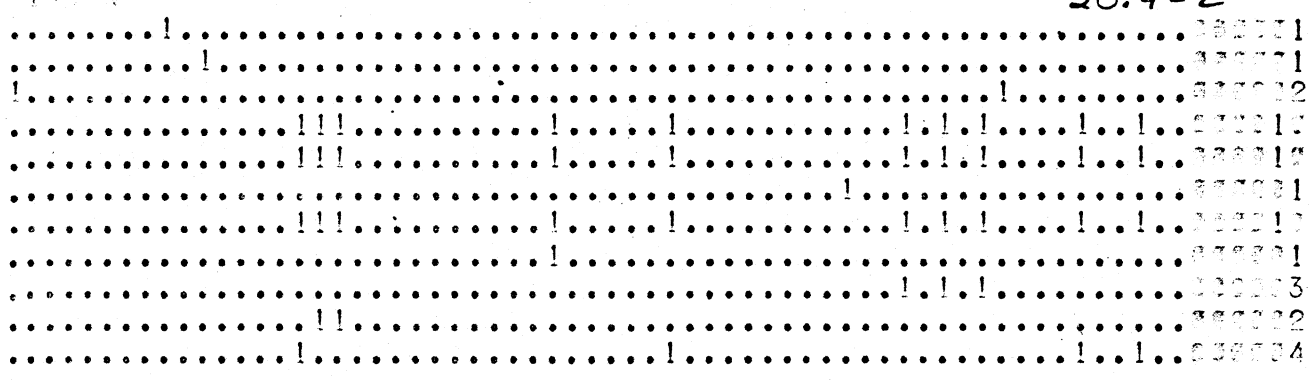
EVANS & SUTHERLAND COMPUTER CORP.
PDP-10 DISPLAY PROCESSOR

BLOCK DIAGRAM
101102-900 2-6-69

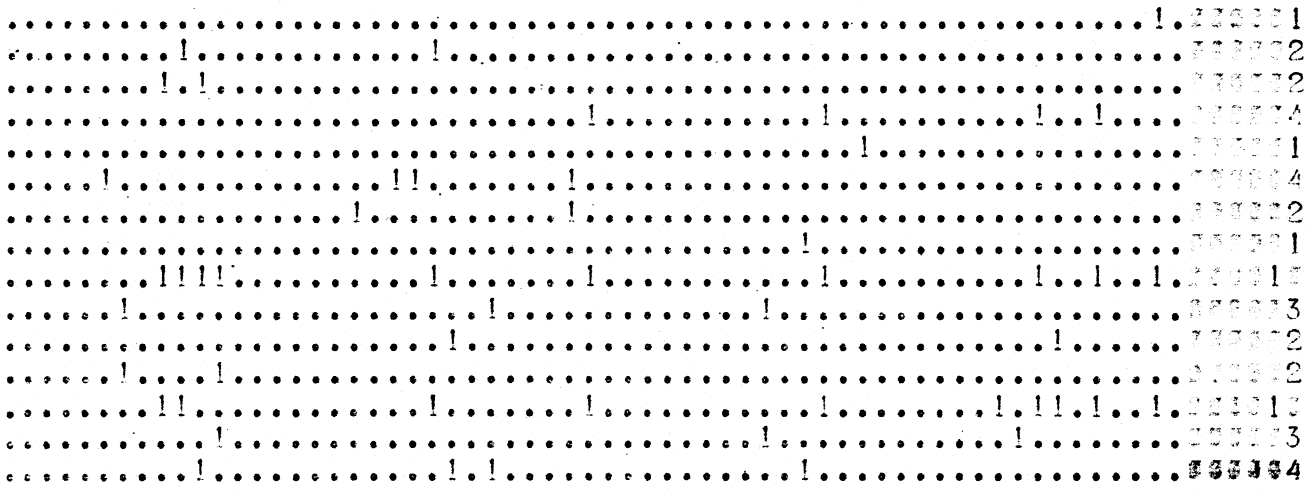
SHEET 1 OF 14

↑SBIG RAM CONTENTS

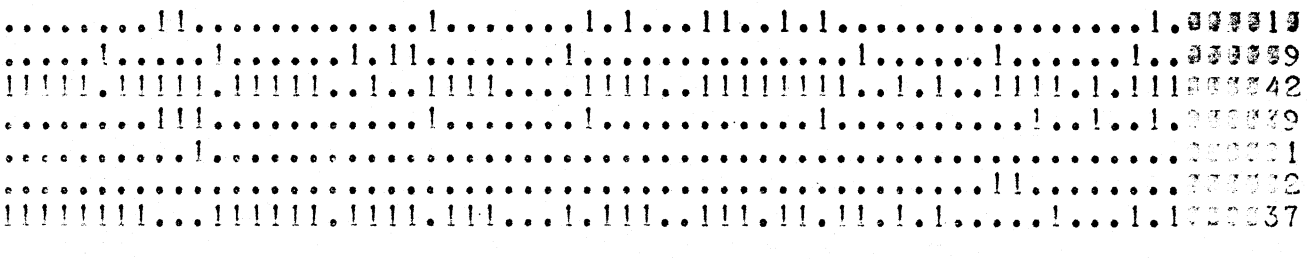




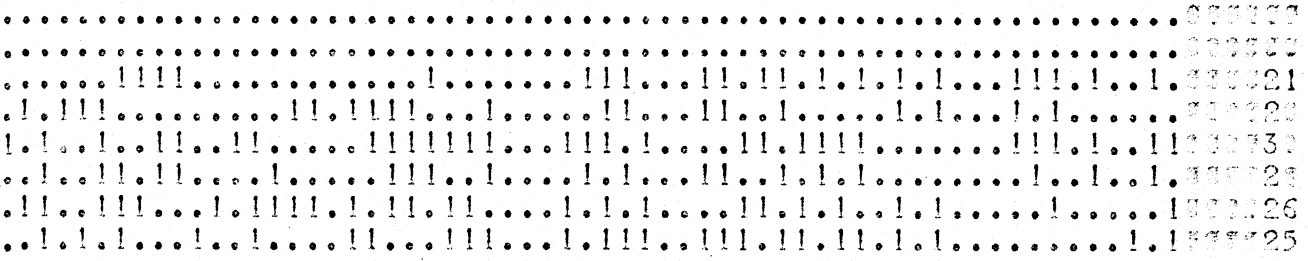
TESTC



JUMPC



JUMPA



WORD LINE DRIVE REQUIRED

CLEA
ICFF
RAFE
PCFE
SPFE
CP3B
DIDI
DIND
STOP
PREP
PAUS
MODE

```
DCIT 000005
PAFF 000012
PCFF 000013
SPFF 000010
IUDI 000010
0PCI 000010
L0ST 000012
SIRT 000013
C0ND 000009
C0NA 000008
C0NB 000010
0PCA 000011
0PCB 000003
0PCC 000004
0PCD 000008
CMND 000008
SIST 000009
SISD 000010
L0RT 000007
L0RU 000006
DINE 000008
DINF 000005
DING 000011
DINH 000012
DINI 000006
DIDV 000007
DDIR 000009
DDIT 000008
DIDU 000009
DINT 000008
WRIT 000007
DSPW 000004
DSPA 000015
PARW 000002
PARA 000013
WARW 000004
WARA 000008
WRIA 000005
WRIB 000006
DATA 000010
DATB 000011
DATC 000009
DATD 000010
DATE 000002
DATF 000015
SISA 000009
SISB 000005
```

```
    .TITLE PROCBA
IS TAPE DEFINES THE COMPONENT PLACEMENT FOR THE PROCESSOR ROM
SE MACROS SET UP TABLES FOR ROM ASSIGNMENTS
 AND BIT SAYS "THIS IS THE FINAL OUTPUT OF THE BIT LINE"
 INCREMENTS 3,4;10,14 ARE APPLIED TO THE INPUTS TO THE
R-INPUT "OR" GATE WHICH PRODUCES THE BIT LINE OUTPUT.
ENSIONS ARE HANDLED BY THE MACROS "CONT", "CONA" AND "CONB".
    .DEFIN BUG2,W
    W+4000
    W-4
    W
    430000
    .ENDM
    .DEFIN BUG4,W
    W+4000
    W
    W+4
    W+10
    W+14
    430000
    .ENDM
    .DEFIN BUG7,W,E
    W+4000
    W
    W+4
    W+10
            W+2014
    CONT E
    430000
    .ENDM
    .DEFIN BUG8,W,E,F
    W+4000
    W-4+2000
    CONT E
    W+2000
    CONT F
    430000
    .ENDM
    .DEFIN BUG10,W,E,F
    W+4000
    W
    W+4
    W+2010
    CONT E
    W+2014
    CONT F
    430000
    .ENDM
    .DEFIN BUG13,W,E,F,G
    W+4000
    W
    W+2004
    CONT E
    W+2010
    CONT F
    W+2014
    CONT G
    430000
    .ENDM
```

```
.DEFIN BUG16,W,E,F,G,H
W+4000
W+2000
CONT E
W+2004
CONT F
W+2010
CONT G
W+2014
CONT H
400000
.ENDM
.DEFIN BUG19,W,E,F,G,H,J
W+4000
W+2000
CONA E,F
W+2004
CONT G
W+2010
CONT H
W+2014
CONT J
400000
.ENDM
.DEFIN BUG22,W,E,F,G,H,J,K
W+4000
W+2000
CONA E,F
W+2004
CONA G,H
W+2010
CONT J
W+2014
CONT K
400000
.ENDM
.DEFIN BUG25,W,A,B,C,D,E,F,G
W+4000  /OUTPUT
W+2000  /FIRST INPUT TO THE OR GATE
CONA A,B        /PUT INPUTS 1-8 ON EXTENDERS
W+2004
CONA C,D
W+2010
CONA E,F
W+2014  /LAST INPUT
CONT G
400000
.ENDM
.DEFIN BUG28,W,A,B,C,D,E,F,G,H
W+4000
W+2003
CONA A,B
W+2004
CONA C,D
W+2010
CONA E,F
W+2014
CONA G,H
400000
.ENDM
```

```
.DEFIN BUG31,W,A,B,C,D,E,F,G,H,I
W+4000
W+2000
CONB A,B,C
W+2004
CONA D,E
W+2010
CONA F,G
W+2014
CONA H,I
400000
.ENDM
.DEFIN BUG34,W,A,B,C,D,E,F,G,H,I,J
W+4000
W+2000
CONB A,B,C
W+2004
CONB D,E,F
W+2010
CONA G,H
W+2014
CONA I,J
400000
.ENDM
.DEFIN BUG37,W,A,B,C,D,E,F,G,H,I,J,K
W+4000
W+2000
CONB A,B,C
W+2004
CONB D,E,F
W+2010
CONB G,H,I
W+2014
CONA J,K
400000
.ENDM
.DEFIN BUG40,W,A,B,C,D,E,F,G,H,I,J,K,L
W+4000
W+2000
CONB A,B,C
W+2004
CONB D,E,F
W+2010
CONB G,H,I
W+2014
CONB J,K,L
400000
.ENDM
.DEFIN CONT,W
W+1000
W
W+4
W+10
W+14
.ENDM
.DEFIN CONA,W,E
W+1000
W
W+4
W+10
```

>

```
        W+2014
        CONT E
        .ENDM
        .DEFIN CONB,W,B,C
        W+1000
        W
        W+4
        W+2010
        CONT B
        W+2014
        CONT C
        .ENDM
        .DEFIN NBIT,N,A,B
        0
        .ASCII +N+
        A
        400000+B
        .ENDM
        .DEFIN BIT,N,A,B
        0
        .ASCII +N+
        A       /MASK
        B       /WORD NUMBER +400000 FOR INVERT HOLE SENSE
        .ENDM
/BUG SYMBOL EQUALITIES
/VALUE OF LAST TWO BITS TELL THE COLUMN NUMBER
        /00=A COLUMN (INNER RIGHT)
        /01=B COLUMN (OUTER LEFT)
        /10=C COLUMN (OUTER RIGHT)
        /11=D COLUMN (INNER LEFT)
A=0
B=1
C=2
D=3
E=20
F=21
G=22
H=23
J=40
K=41
L=42
M=43
N=60
P=61
Q=62
R=63
S=100
T=101
U=102
V=103
W=120
X=121
Y=122
Z=123
AA=140
BB=141
CC=142
DD=143
EE=160
FF=161
>
```

```
GG=162
HH=163
JJ=200
KK=201
LL=202
MM=203
EEU=164
EEL=174
FFU=165
FFL=175
GGU=166
GGL=176
HHU=167
HHL=177
JJU=204
JJL=214
KKU=205
KKL=215
LLU=206
LLL=216
MMU=207
MML=217
        .GLOBL BOARD1,BOARD2,BOARD3,BOARD4
        .GLOBL BOARD5,BOARD6,BOARD7,BOARD8
/HERE ARE THE BOARD DEFINITIONS
BOARD1 1        /TENS DIGIT
        1       /UNITS
        NBIT JUC3,20,7
        BUG22 W,A,C,E,G,J,L
        NBIT JUC7,1,7
        BUG28 X,B,D,F,H,K,M,P,R
        BIT REC2,1000,1
        BUG4 CC
        BIT PAC4,40,2
        BUG4 FF
        BIT PKC1,4000,3
        BUG4 HH
        BIT PKC5,200,3
        BUG4 KK
        BIT PKC6,100,3
        BUG4 MM
        BIT PKC8,20,3
        BUG4 EE
        BIT PKI1,2,3
        BUG4 GG
        400001
BOARD2 1        /TENS
        2       /UNITS
        BIT JUA5,10,10
        BUG31 AA,A,C,E,G,J,L,N,Q,Y
        BIT JUA7,2,10
        BUG28 BB,B,D,F,H,K,M,P,R
        BIT REI1,1,1
        BUG13 CC,S,U,W
        BIT FLC2,1000,4
        BUG4 GG
        BIT FLC8,10,4
        BUG4 FF
        BIT IOC9,4,5
        BUG4 HH
>
```

```
        BIT IO11,1,5
        BUG4 JJ
        BIT TSC3,4000,6
        BUG4 LL
        400001
BOARD3  1         /TENS
        3         /UNITS
        BIT JUA8,1,10
        BUG25 AA,A,C,E,G,J,L,N
        BIT JUA6,4,10
        BUG22 BB,B,D,F,H,K,Z
        BIT PAC9,1,2
        BUG19 DD,M,P,R,T,V
        BIT TSC5,1000,6
        BUG4 HH
        BIT TSC9,40,6
        BUG4 EE
        BIT TS14,1,6
        BUG4 GG
        BIT TSC7,200,6
        BUG4 LL
        BIT FLC7,20,4
        BUG2 KKU
        BIT FLC9,4,4
        BUG2 KKL
        BIT FL10,2,4
        BUG4 JJ
        BIT FL11,1,4
        BUG2 MML
        400001
BOARD4  1         /TENS
        4         /UNITS
        BIT REC1,2000,1
        BUG16 AA,A,C,E,G
        BIT PAC6,10,2
        BUG10 CC,J,L
        BIT PK12,1,3
        BUG10 X,P,K
        BIT IOC4,200,5
        BUG10 Z,R,M
        BIT IOC7,20,5
        BUG10 BB,B,D
        BIT TS12,4,6
        BUG10 DD,F,H
        BIT IOC1,2000,5
        BUG2 KKU
        BIT IOC2,1000,5
        BUG2 KKL
        BIT IOC8,10,5
        BUG2 MMU
        BIT IO10,2,5
        BUG2 MML
        BIT TSC1,20000,6
        BUG2 JJU
        BIT TSC2,10000,6
        BUG2 JJL
        BIT TSC4,2000,6
        BUG2 LLU
        BIT TSC6,400,6
        BUG2 LLL
>
```

```
        BIT TS10,20,6
        BUG2 FFU
        BIT TS11,10,6
        BUG2 FFL
        BIT TS13,2,6
        BUG4 Y
        BIT JUC5,4,7
        BUG2 EEU
        BIT JUC6,2,7
        BUG2 EEL
        BIT IOC3,400,5
        BUG2 GGU
        BIT IOC6,40,5
        BUG2 GGL
        BIT TSC0,40000,6
        BUG2 HHL
        400001
BOARD5  1       /TENS
        5       /UNITS
        BIT JUC4,10,7
        BUG10 AA,A,C
        BIT REC4,200,1
        BUG2 FFU
        BIT REC5,100,1
        BUG2 FFL
        BIT REC7,20,1
        BUG2 HHU
        BIT REC8,10,1
        BUG2 HHL
        BIT PAC2,200,2
        BUG2 KKU
        BIT PKC2,2000,3
        BUG2 KKL
        BIT PKC3,1000,3
        BUG2 MMU
        BIT PKC4,400,3
        BUG2 MML
        BIT PKC7,40,3
        BUG2 EEU
        BIT PKC9,10,3
        BUG2 EEL
        BIT PK10,4,3
        BUG2 GGU
        BIT FLC1,2000,4
        BUG2 GGL
        BIT FLC3,400,4
        BUG2 JJU
        BIT FLC4,200,4
        BUG2 JJL
        BIT FLC5,100,4
        BUG2 LLU
        BIT FLC6,40,4
        BUG2 LLL
        BIT REC3,400,1
        BUG7 CC,G
        BIT RE10,2,1
        BUG10 Y,E,J
        BIT JUC1,100,7
        BUG10 BB,B,F
        BIT JUC2,40,7
>
```

```
        BUG1Ø DD,D,H
        BIT IOC5,1ØØ,5
        BUG1Ø W,N,Q
        BIT RECS,A,1
        BUG1Ø X,P,K
        BIT TSCS,1ØØ,6
        BUG1Ø Z,R,M
        4ØØØØ1
BOARD6  1           /TENS
        6           /UNITS
        BIT JUA4,2Ø,1Ø
        BUG22 BB,B,D,F,H,K,M
        BIT JUA3,4Ø,1Ø
        BUG22 AA,A,C,E,G,J,L
        BIT PAC3,1ØØ,2
        BUG7 DD,R
        4ØØØØ1
BOARD7  Ø
BOARD8  Ø
        .END
PXEND OF FILE REACHED BY:
P 6Ø
>
```